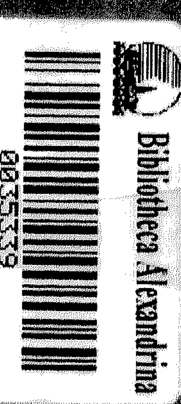
تطبيقات نظم إدارة قواعد البيانات



طفى رضا عبد الوهاب — ا.د. محمد فهمى طلبه

لا. الدين محمد فهمى — ا.د. عمرو جنيد

ى محمد إسماعيل — ا.د. محمد على الشرقاوى



٧

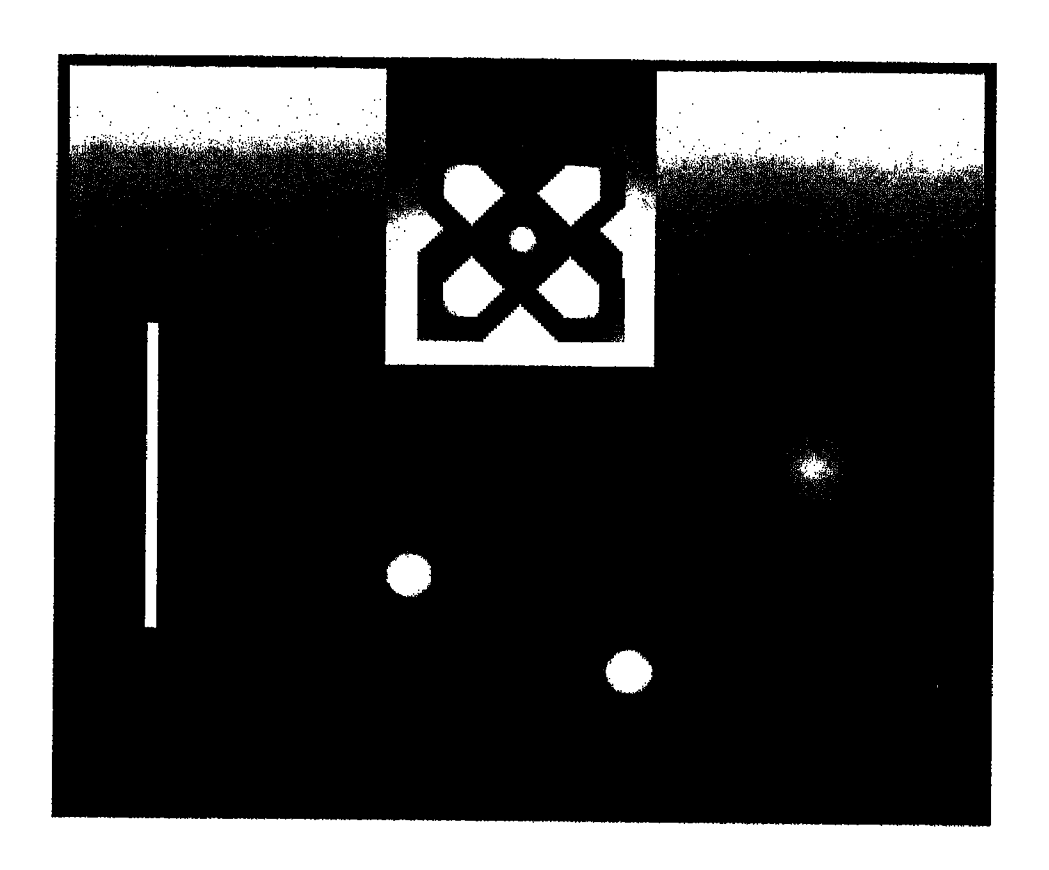
موسوعة دلتا كمبيوتر



تطبيقات نظم إدارة قواعد البيانات



تطبيقات نظم إدارة قواعد البيانات



م. مصطفى رضا عبد الوهاب
د. علاء الدين محمد فهمى
مصطفى محمد إسماعيل

أ.د. محمد فهمى طلبه
أ.د. عمرو جنيد
أ.د. محمد على الشرقاوى

٧

موسوعة دلتا كمبيوتر

رقم الإيداع : ٩١/٢٨٨٥

مطابع المكتب المصرى الحديث
MODERN EGYPTIAN PRESS

تقـــديم

ان السؤال الدائر بين المتخصصين فى مجال الحاسبات عن كون تخطيط البرامج على وجه الخصوص وبعض جوانب علوم الحاسب عامة تقع فى اطار العلم (Science) أو فى اطار الفن (Art) . وبدون الخوض فى تفاصيل تعريف كل منهما من ناحية الأسس القائم عليها وعلاقة ذلك بالانسان فانه يوجد شبه اتفاق بين كل الخبراء فى أن تخطيط البرامج وان كان يرتكز على العديد من الأسس العلمية التى يجب الالمام بها الا أنه يحتاج الى العديد من الجوانب المرتبطة بالفن والتى تتأثر بخبرة مخطط البرامج وقدرته الابداعية بالاضافة الى شخصيته .

ومما لاشك فيه أن البرنامج الواحد - الذى يؤدى مهمة معينة للوصول الى هدف أو أهداف محددة - اذا ما تم تنفيذه بواسطة العديد من مخططى البرامج فانه يصعب أن تتفق خطوات التنفيذ لأى منهم مع الاخر رغم أن النظام النهائى يؤدى نفس المهمة . ولا يعتبر الاختلاف فى النظم المنتجة ناشئا فقط من التباين فى الخلفية العلمية لمخططى البرامج بقدر ما يعبر عن التباين فى خبرتهم وشخصيتهم بالاضافة الى قدرتهم الابداعية .

ولا خلاف فى أن الخبرة والقدرة الابداعية يمكن أن تصقل بواسطة الممارسة بعد التقليد . ومن هذا البعد كانت فكرة هذا الكتاب الذى يعتبر من أحد أهدافه تمكين مخطط البرامج - الملم بكل الجوانب العلمية لهندسة تخطيط البرامج والمتمكن من أدوات البرمجة المستخدمة - من متابعة بعض النماذج للحلول المقترحة لمشاكل معينة بحيث تمكنه بصورة أو بأخرى من تقليدها فى أحد الاطارات الاخرى المشابهة .ومع التباين بين النظم التى تم مناقشتها يستطيع القارىء اكتساب بعض الخبرات فى العديد من المجالات المطروحة فى هذا الكتاب .

ويمكن للقارىء أن ينتقل من مرحلة الاستيعاب الى مرحلة التقليد ثم الى مرحلة التعديل يليها مرحلة التطوير ثم ينطلق الى مرحلة الابداع . وقد اختار هذا الكتاب

أحد أدوات البرمجة المتميزة وهى نظم برامج عائلة DBASE . وتنتمى الى هذه العائلة نظم برامج DBASE III+ , DBASE IV , FOXBASE , FOXPRO ولابد للقارىء عند الاطلاع على هذا الكتاب أن يكون ملما بأوامر وأساليب البرمجة المستخدمة فى لغة هذه العائلة . وجدير بالذكر أن كتاب نظم ادارة قواعد البيانات "الجزء الأول" و "الجزء الثانى" من موسوعة "دلتا كمبيوتر" لتكنولوجيا وعلوم الحاسب تمكن القارىء المبتدىء من التعرف على أوامر وأساليب البرمجة باستخدام أى من نظم برامج عائلة DBASE .

وفى مرحلة الاستيعاب يمكن للقارىء أن يركز فى مضمون النظم الموضحة فى أجزاء وفصول الكتاب المختلفة ، وأن يحدد وظيفة كل جزء من هذه الأجزاء ومن ثم التعرف على بعض الأفكار المطروحة والاطار العام لتصميم النظام وهيكله البنائى .

اما فى مرحلة التقليد فانه يمكن للقارىء أن يستخدم نفس المنهج المحدد فى هذا الكتاب لبناء نظم مشابهة . وعندما يتمكن القارىء من كل مفاتيح عملية تخطيط البرامج يستطيع اجراء التعديلات المختلفة على الامثلة المحددة فى هذا الكتاب بما يتلائم مع احتياجات نظامه وبما يمكنه من الوصول الى الهدف المطلوب . عند هذه المرحلة يكون القارىء قد بنى خبرة جيدة ومتميزة فى انتاج البرامج باستخدام نظم عائله DBASE يتمكن بعدها من تطوير العديد من البرامج المشابهة والغير مشابهة للنظم المذكورة فى هذا الكتاب ثم يلى ذلك مرحلة الابداع حيث يستطيع القارىء ايجاد أساليب مختلفة تعبر عن شخصيته وتميز أسلوبه وقدرته على الابتكار .

ويتكون الكتاب من ست وعشرين فصلا مقسمة الى خمسة أجزاء . الجزء الأول منها مراجعة شاملة لنظم برامج عائلة DBASE . أما الجزء الثانى فيوضح نظام معلومات لشئون الطلبة الذى يصلح أن يستخدم فى العديد من المدارس وبعض كليات الجامعة بطريقة مبسطة وسهلة . ويعتبر الجزء الثالث أحد البرامج المحاسبية الشائعة الاستخدام وهو يوضح نظام المخازن . أما الجزء الرابع فهو متعلق بنظام حسابات العملاء وهو أيضا من البرامج المحاسبية المستخدمة فى العديد من المؤسسات التجارية والصناعية . أما الجزء الخامس فانه يضيف بعض الأساليب المتقدمة لتخطيط البرامج والتى يمكن أن يستفيد منها مخطط البرامج فى رفع كفاءته وقدرته على

استخدام نظم برامج عائلة DBASE .

ويعتبر هذا الكتاب أحد المحاولات الجيدة التى يمكن ان توضح للدارس بعض النظم التطبيقية الشائعة الاستخدام . ومن هذا البعد - بالاضافة الى ماسبق ذكره - فان هذا الكتاب يعتبر اضافة حقيقية للمكتبة العربية فى مجال تكنولوجيا وعلوم الحاسب .

ا. د. محمد فهمى طلبه



محتويات الكتاب

مسلسل — الموضوع

الفصل الرابع عشر : "برنامج تشغيل الملف الرئيسى"

الفصل الخامس عشر : "برنامج تشغيل ملف المبيعات"

الفصل السادس عشر : "برنامج تشغيل ملف الاضافة"

مسلسل	الموضوع	رقم الصفحة
٢٠ - ٥ - ٢	تعديل ملف الصرف (EdChrg.prg)	٢٩٧
٢٠ - ٥ - ٣	تعديل ملف السداد (EdPay.prg)	٣٠١
	الفصل الحادى والعشرون : "تقارير برنامج حسابات العملاء"	٣٠٥
٢١ - ١	برنامج قائمة التقارير الرئيسية (ARPrint.prg)	٣٠٧
٢١ - ٢	ملف الخطوات (BillProc.prg)	٣٠٩
٢١ - ٣	برنامج الفواتير الشهرية (Bills.prg)	٣١٣
٢١ - ٤	برنامج اختبار الحالة (ARStat.prg)	٣١٦
٢١ - ٥	برنامج التقارير المختصرة والزمنية (AgeSumm.prg)	٣١٧
٢١ - ٦	التقارير التاريخية (ARHist.prg)	٣٢٢
	الفصل الثانى والعشرون : "التحديدث الشهرى للنظام"	٣٢٩
	الفصل الثالث والعشرون : "برنامج التكامل بين حسابات العملاء والمخازن"	٣٣٩
	الجزء الخامس : بعض الأدوات المتقدمة	٣٤٥
	الفصل الرابع والعشرون : "برنامج كتابة الشيكات"	٣٤٩
	الفصل الخامس والعشرون : "برنامج اختيار الألوان"	٣٥٩
	الفصا السادس والعشرون : "برنامج تحريك العمود الضوئى"	٣٦٥

الفصل الأول

مقدمة



هذا الكتاب يمثل التطبيق العملي للكتاب السابق على نظم المعلومات الشائعة الاستخدام مثل نظم معلومات شئون الطلبة (Cadets)، ونظم المخازن (Inventory)ونظم حسابات العملاء (Accounts Receivable). وهو يركز على أحدث الوسائل (Techniques) التي تستخدم في كتابة البرامج القوية ذات الكفاءة العالية وسرعة التشغيل الكبيرة .

والكتاب لايكتفي بعرض البرامج ولكنه يشرح كل برنامج خطوة خطوة حتى يستوعب القارىء البرامج . كما أنه يراعي استخدام معظم أوامر عائلة (DBase)والدوال الخاصة بها واستخدام كل البدائل الممكنة حتى يصبح القارىء ملما بجميع امكانيات البرنامج ، كما تتوفر لديه القدرة على كتابة برامج كاملة قابلة للتنفيذ . كما توفر مؤسسة دلتا أقراصا تحتوى على هذه البرامج ، يستطيع القارئ - الذى لايجد وقتا كافيا لكتابتها - استخدامها مباشرة من الأقراص مع امكانية تتبع سطور البرنامج و مراجعتها قبل تنفيذها .

وهذا الكتاب كما سبق الايضاح هو استكمال للكتاب السابق ، لذلك فمن المفيد قراءة الكتاب الأول حتى يستطيع القارىء استيعاب البرامج الموجودة . ويستطيع القارىء كتابة هذه البرامج على الحاسب وتنفيذها حيث أن هذا يكسبه الخبرة المطلوبة لكتابة أى برامج أخرى .

والكتاب ينقسم الى خمسة أجزاء ، الجزء الأول عبارة عن مراجعة شاملة لبرامج عائلة (DBase) والأوامر المستخدمة فيها . مع شرح لطرق كتابة البرامج وبعض الوسائل التى تزيد من كفاءتها وسرعة تنفيذها .

والجزء الثاني يشرح برنامج نظام معلومات شئون الطلبة (Cadets) الذى يمثل نوعا من البرامج التي تتعامل مع قاعدة بيانات واحدة (Single Database) . وهو برنامج يصلح للمبتدئين حيث أنه يركز على أساسيات كتابة البرامج التي تعتمد على القوائم فى تشغيلها بواسطة المستخدم (User Friendly) . كما يركز أيضا على تصميم شاشات الادخال وتصميم التقارير .

والجزء الثالث يشرح برنامج المخازن (Inventory) الذى يوضح أساسيات التعامل مع عدة ملفات قواعد بيانات وربط هذه الملفات وفتحها من مناطق عمل مختلفة (Work Areas) .

والجزء الرابع يشرح برنامج حسابات العملاء (Accounts Receivable) الذى يضيف امكانيات متقدمة للتعامل مع الملفات المرتبطة (Related) وتحديثها واستخدام الملفات التاريخية (History Files) للاحتفاظ بالبيانات القديمة .

والجـــــــزء الخـــامـس يضـــــيـف بعـض الـوسـائـل المتقـــدمـة (Advanced Techniques) التى يمكن لمخطط البرامج استخدامها مع أى برنامج لزيادة كفاءته .

ويلاحظ أن الكتاب يتدرج فى درجة صعوبة البرامج حتى يصل بالقارىء فى نهاية الكتاب الى الخبرة الكافية والقدرة على التعامل مع أعقد نظم المعلومات .

ملاحظة

البرامج المشروحة فى هذا الكتاب تعمل على جميع برامج عائلة (DBase) مثل (DBase IV) ، (FoxBase +) ، (FoxPro) . كما ان الاوامر المستخدمة فى كتابة هذه البرامج هى نفس الأوامر المستخدمة فى برامج عائلة (DBase) الأخرى . ارجع الى الكتابين الأول و الثانى لمراجعة أوامر عائلة (DBase) .

١ - ١ مشيرة النقطة (Dot Prompt)

كما سبق الايضاح فى الكتابين السابقين فان برنامج (DBase III+) ينقسم الى جزأين رئيسيين هما برنامج المساعد (Assistant) ومشـيرة النقـطة (Dot Prompt) . واستخدام مشيرة النقطة هو الأساس فى كتابة البرامج .

ويتم عرض مشيرة النقطة عن طريق الضغط على مفتاح الهروب (Esc) عند بدء تشغيل برنامج (DBase III+) وظهور قوائم المساعد (Assistant) . ويمكن عرض مشيرة النقطة مباشرة بعد تحميل البرنامج . ولتنفيذ ذلك يتم اجراء تعديل بسيط فى ملف المواصفات (Config.DB) وذلك بالغاء السـطرين التاليين من الملف .

```
STATUS ON
COMMAND = ASSIST
```

١- ٢ تعديل مواصفات النظام (Config.sys)

كما سيلاحظ القارىء فيما بعد ، أن بعض البرامج التى سيتم تصميمها خلال هذا الكتاب تحتوى على العديد من الملفات التى يلزم فتحها فى نفس الوقت ، فاذا

كان ملف مواصفات النظام (Config.sys) لايحتوى على الأمر الذى يسمح باستخدام عدد كبير من الملفات ، فان من المتوقع عند تشغيل أى برنامج كبير ظهور الرسالة التالية :

Too many files are open

وللتغلب على ذلك يتم تعديل ملف مواصفات النظام (Config.sys) باستخدام برنامج (EDLIN) أو أى برنامج معالجة كلمات مع الشكل غير الوثائقى (Non Document Format) . ويتم كتابة السطرين التاليين .

```
FILES = 20
BUFFERS = 15
```

ويجب بعد ذلك اطفاء الجهاز واعادة تشغيله من جديد حتى يتم تحميل ملف المواصفات وتخصيص الملفات (Files) ومخازن الذاكرة (Buffers) المطلوبة للبرامج .



١

الجزء الأول

مراجعة كتاب (نظم ادارة قواعد البيانات)

* أنواع البرامج
* البحث السريع
* خطوات تصميم النظام
* كتابة البرامج
* وسائل التصحيح



الفصل الثاني

أنواع البرامج


Converted by Tiff Combine - (no stamps are applied by registered version)

من المهم قبل البدء في تصميم البرنامج ، تحديد نوع هذا البرنامج ، اذا كان من البرامج التي تتعامل مع ملف قاعدة بيانات واحد (Single Database) مثل برنامج شئون الطلبة (Cadets) كما سيتم الايضاح ، أو من البرامج التي تتعامل مع عدة ملفات مرتبطة (Relational Databases) ، أو من البرامج التي تتعامل مع ملف رئيسي وملفات حركة (Master/Transaction Databases) . لذلك سوف يتم القاء الضوء على كل نوع من هذه الأنواع الثلاثة قبل تصميم البرامج التي تنتمي الى كل نوع .

٢ - ١ التعامل مع قاعدة بيانات واحدة

البرنامج في هذه الحالة يعتبر أبسط أنواع البرامج . حيث يكفي تصميم ملف قاعدة بيانات واحد، ويتعامل البرنامج دائما مع هذا الملف . لذلك لاتكون هناك حاجة الى فتح عدة ملفات واستخدام عدة مناطق عمل (Work Areas) في نفس الوقت . وهذا يسهل التحكم في البرنامج ومتابعة تنفيذه .

وعادة يتم انشاء ملف قاعدة البيانات قبل البدء في كتابة البرنامج . ويتم ذلك بكتابة السطر التالي من مشيرة النقطة (Dot Prompt).

```
CREATE Cadets
```

حيث يكون الملف (Cadets.dbf) هو ملف قاعدة البيانات المطلوب انشاؤه .

وفي هذه الحالة تظهر قائمة تحديد الحقول التي يتم من خلالها تحديد اسم كل حقل ونوعه وعرضه وعدد الأرقام العشرية ان وجدت .

كما يمكن انشاء ملف الفهرس من مشيرة النقطة أيضا كالآتي مثلا :

```
USE Cadets
INDEX ON Name TO Name
```

في هذه الحالة يتم ترتيب السجلات بناء على الترتيب الهجائي لأسماء الطلبة . كما يمكن فهرسة الملفات بناء على حقلين ، حيث يكون الحقل الأول هو الحقل الرئيسي الذي يتم الترتيب بناء عليه . وذلك كالآتي مثلا :

```
USE Cadets
INDEX ON Name + Class TO Name
```

وعندما يراد فتح ملف الفهرس يتم كتابة الآتى مثلا :

```
USE Cadets INDEX Name
```

وهذا يؤدى الى فتح ملف قاعدة البيانات وملف الفهرس الخاص به . كما يمكن فتح ملف الفهرس وحده عندما يكون قد سبق فتح ملف قاعدة البيانات وذلك كالآتى :

```
SET INDEX TO Name
```

٢ - ٢ قواعد البيانات المرتبطة (Relational Databases)

قواعد البيانات المرتبطة هى الملفات التى تكون مرتبطة ببعضها بناء على حقل مشترك . وهى تفيد بصفة خاصة في التخلص من أى تكرار للبيانات . وهذا يؤدى الى تقليل المساحة التخزينية المستخدمة بالاضافة الى زيادة سرعة التشغيل .

فمثلا عند تصميم ملف قاعدة بيانات لحسابات العملاء (AR.dbf) ، يمكن تصميمه كالآتى :

Structure for database :C: AR.dbf

Field	Field Name	Type	Width	Dec
1	BILL_DATE	Date	8	
2	AMOUNT	Numeric	9	2
3	VENDOR	Character	20	
4	ADDRESS	Character	20	

وهذا يعتبر تصميما سيئا للملف ، حيث قد يكون هناك بائعون (Vendors)لهم مئات الفواتير بتواريخ مختلفة . وفى هذه الحالة يتم تكرار أسماء هؤلاء البائعين وعناوينهم .

ولكن التصميم الأفضل هو تقسيم الملف الى ملفين منفصلين ، الملف الأول هو الملف (AR1.dbf) مع تخزين البيانات الشخصية للبائعين مثل الاسم والعنوان فى

ملف آخر (AR2.dbf) .

فمثلا يمكن انشاء الملف (AR1) كالآتى :

Structure for database :C: AR1.dbf

Field	Field Name	Type	Width	Dec
1	BILL_DATE	Date	8	
2	AMOUNT	Numeric	9	2
3	VEND_CODE	Character	5	

كما يمكن انشاء الملف (AR2.dbf) كالآتى :

Field	Field name	Type	Width	Dec
1	VEND_CODE	Character	5	
2	VENDOR	Character	20	
3	ADDRESS	Character	20	

ويلاحظ أن الحقل الوحيد المكرر هو حقل (Vend_Code) . وهو الحقل الذى يستخدم فى ربط الملفين .

ولربط هذين الملفين يتم فهرسة أحدهما على الحقل المشترك(Vend_Code) كالآتى :

```
USE AR2
INDEX ON Vend_Code TO Vendor
```

ثم يتم فتح كل ملف فى منطقة عمل مختلفة (Work Area) . ويتم انشاء العلاقة بين الملفين باستخدام الأمر (SET RELATION TO) . وذلك يتضح من السطور التالية :

```
SELECT 1
USE AR1
SELECT 2
USE AR2 INDEX Vendor
SELECT 1
SET RELATION TO Vend_Code INTO AR2
```

وعندما يراد عرض أى حقل من حقول الملف (AR2) يستخدم الرمز (B ->) . حيث أن (B) هنا يمثل الاسم المرادف (Alias) لمنطقة العمل رقم (2) ، وسيتم شرح ذلك بالتفصيل عند شرح نظام حسابات العملاء (Accounts Receivable).

٢ - ٣ نظام الملف الرئيسي وملفات الحركة

هذا النوع من البرامج يستخدم عادة مع نظم المخازن (Inventory) والمكتبات (Libraries) والبنوك (Banks) ، حيث يكون هناك ملف بيانات رئيسي (Master File) يحتوى على بيانات كل حركة اضافة أو صرف . ويتم تحديث بيانات الملف الرئيسي (Master) من ملفات الحركة (Transaction). فمثلا نظام المخازن في أبسط صورة يحتوي على ملف رئيسي واحد وملفين حركة . ويكون هيكل النظام كما هو موضح من الشكل (٢ - ١) .

والملف الرئيسي في هذا النظام مثلا يحتوى على البيانات الحالية عن المخزون الفعلى (On hand stock). كما يحصل على بيانات كل حركة صرف أو توريد من ملفات الحركة حتى يعطي دائما الموقف الحالى للمخزون . وفائدة هذا النظام أنه يعطى المستخدم الموقف الحالى فى كل لحظة ، كما يتابع حركة الصرف والتوريد.

ويلاحظ وجود حقل مشترك بين الملف الرئيسي وملفات الحركة وهو حقل رقم الجزء (Part_No) . وهو حقل منفرد (Unique) يستخدم فى ربط الملفات الثلاثة . ويستخدم الأمر (UPDATE) في تحديث الملف الرئيسي من ملفات الحركة . ولتنفيذ ذلك يجب أولا فهرسة الملفات على حقل رقم الجزء (Part_No).

Inventory.dbf
Part Number
Part name
Amount in stock
Price
Reorder point
Amount on order
Due date

Sales.dbf
Part number
Part name
Quantity sold
Selling price
To whome sold
Date sold

Purchase.dbf
Part number
Quantity received
Purchase price
Vendor name
Date received

اشكل (٢ - ١)

فمثلا لتحديث الملف الرئيسى (Master) من ملف المبيعات (Sales) يجب أولا كتابة السطور التالية :

```
USE Inventory
INDEX ON Part_No TO Master
USE Sales
INDEX ON Part_No TO Sales
```

ثم يتم فتح كل ملف فى منطقة عمل مختلفة (Work Area) باستخدام الأمر (SELECT) ، ثم يستخدم الأمر (UPDATE) فى نقل البيانات من ملف المبيعات الى الملف الرئيسى . وذلك كالآتى :

```
SELECT 2
USE Sales INDEX Sales
SELECT 1
USE Inventory INDEX Master
UPDATE ON Part_No FROM Sales REPLACE On_Hand WITH ;
On_Hand - B -> Qty
```

ملاحظة

يجب ملاحظة أنه عند زيادة الأمر عن طول السطر على الشاشة يتم استخدام حرف الفاصلة المنقوطة (;) . ويجب أن يعرف مخطط البرامج وظيفة الفاصلة المنقوطة فى هذه الحالة حتى لايحدث خطأ في كتابة الأمر، حيث أن هذا الحرف يؤدى الى ضم السطر التالي الى السطر الجاري كتابته . ولذلك يراعي عندما يكون مطلوبا وجود مسافة خالية (Space) ترك هذه المسافة قبل كتابة هذا الحرف . لأن هذا الحرف يضم السطر التالي دون أى مسافات .

ويلاحظ هذا فى السطر الأخير من الأوامر السابقة . وهذا السطر يؤدى الى تحديث الملف الرئيسى (Master.dbf) من ملف المبيعات (Sales.dbf) عن طريق استبدال محتويات حقل الكمية الفعلية (On_Hand) بنفس المحتويات مطروحا منها كمية الصنف التى تم بيعها (B -> Qty) والتى يتم الحصول عليها من ملف المبيعات فى منطقة العمل رقم (٢) .

ويمكن اجراء عملية التحديث من ملف المشتريات (Purchases) بنفس الطريقة . والفارق الوحيد هو اضافة الكمية المشتراة (B -> Qty) من ملف المشتريات فى منطقة العمل رقم (٢) الى الكمية الفعلية . وذلك كالآتى :

```
UPDATE ON Part_No FROM Purchase REPLACE On_Hand WITH ;
On_Hand + B -> Qty
```

وسوف يتم شرح كل الوسائل المتاحة لكتابة البرامج الخاصة بهذا النظام فى الجزء الخاص بدراسة برنامج المخازن (Inventory) .

٢ - ٤ استخدام حقول الملاحظات

هناك بعض قواعد البيانات التى تحتاج الى استخدام حقل للملاحظات . وأوضح مثل لذلك قاعدة بيانات المكتبة (Library) . حيث يراد مثلا انشاء ملف يحتوى على بيانات بالأبحاث أو الكتب الخاصة بكل مؤلف . هذا الملف يمكن أن يحتوى على حقول اسم المؤلف ، عنوان البحث أو الكتاب ، تاريخ النشر ، اسم الناشر ، الموضوعات (Topics) ، والملخص (Abstract) . ويمكن تصميم هيكل الملف (Structure) فى هذه الحالة بطريقتين . فى الطريقة الأولى نسمى الملف مثلا (Lib1.dbf) وندخل الحقول الخاصة به كالآتى :

Structure for database :C: Lib1.dbf				
Field	Field name	Type	Width	Dec
1	AUTHOR	Character	20	
2	TITLE	Character	20	
3	PUB	Character	20	
4	DATE	Date	8	
5	TOPICS	Character	60	
6	ABSTRACT	Character	254	

شكل (٢ - ٢)

ويلاحظ فى هذه الحالة أنه تم تخصيص (٢٥٤) حرفا للملخص (Abstract) . وذلك لأن هذا العدد يمثل الحد الأقصى لعرض الحقل الحرفى . وهذا العرض يسمح بكتابة سطور معدودة لاتزيد عن أربعة سطور .

ولكن قد يكون مطلوبا كتابة ملخص كبير يزيد عن صفحة لكل كتاب مثلا . وفى هذه الحالة يتم استخدام نوع آخر من الحقول يسمى حقل الملاحظات (memo field) . وهذا النوع من الحقول يسمح بكتابة حتى (٤٠٠٠) حرفا فى الحقل الواحد . ويمكن زيادة عدد الحروف أكثر من ذلك عن طريق استخدام معالج كلمات (Word Processor) آخر غير المستخدم مع برنامج (DBase III+) .

والشكل التالى يوضح هيكل الملف (Lib2.dbf) بعد استخدام حقل الملاحظات (memo field) .

Structure for database Lib2.dbf				
Field	Field name	Type	Width	Dec
1	AUTHOR	Character	20	
2	TILTE	Character	20	
3	PUB	Character	20	
4	DATE	Date	8	
5	TOPICS	Character	60	
6	ABSTRACT	Memo	10	

شكل (٢ - ٣)

ويلاحظ هنا أن البرنامج يخصص الرقم (١٠) آليا لعرض حقل الملاحظات (Abstract) رغم أنه يسمح فعليا بتخزين حتى (٤٠٠٠) حرف . وهذا لأن ما يكتب في هذا الحقل يخزن فعليا في ملف قاعدة بيانات مساعد (Auxiliary) يكون امتداده (.dbt) وليس (.dbf) . ولكن مكان هذا الحقل فقط هو الذى يتم تخزينه في ملف قاعدة البيانات الأصلى .

ولادخال بيانات في حقل الملاحظات يتم ذلك من خلال شاشة الادخال حيث يتم وضع المؤشر على حقل الملاحظات والضغط على مفتاحي (Ctrl-PgDn) . وفي هذه الحالة يتم مسح الشاشة وتصبح الشاشة جاهزة لادخال بيانات هذا الحقل. وعند الانتهاء من الكتابة يتم الضغط على مفتاحي (Ctrl-PgUp) أو مفتاحي (Ctrl-W) أو مفتاحي (Ctrl-end) لتخزين بيانات هذا الحقل والرجوع الى شاشة الادخال مرة ثانية .

ويلاحظ عند استخدام الأمر(LIST) في عرض بيانات الحقول عدم ظهور محتويات حقل الملاحظات ولكن تظهر كلمة (memo) . ولكن عندما يراد عرض محتويات هذا الحقل يتم تحديد اسم الحقل مع الأمر(LIST) كالآتي مثلا :

```
LIST OFF Author,Title,Pub,Date,Abstract
```

وفي هذه الحالة يظهر الآتي :

Garton J. T. Decision Support Systems Byte Magazine 03/01/85

This article discusses automated Decissions , Support Systems used in modern business Mini and Microcomputer systems. It includes a review and comparison of several currently available systems , both as support , knowledge Maker , Mind games and Decisive.

Franklin B.W. Automated MBO MBO Monthly 05/01/85

Describes several automated systems that support Management by objectives (MBO). The basic theory of MBO is discussed, the several MBO systems are reviewed.

شكل (٢ - ٤)

ويمكن التحكم في عرض الكتابة في حقل الملاحظات عن طريق كتابة الأمر (SET MEMOWIDTH) . فمثلا لتحديد العرض (٤٠) لحقل الملاحظات يتم كتابة الأمر التالى :

```
SET MEMOWIDTH TO 40
```

كما يمكن التحكم في مكان ظهور الملاحظات وشكلها من خلال ملف الأوامر (Command File) . فمثلا لعرض نفس الملاحظات السابقة بطريقة أوضح يمكن كتابة البرنامج التالى :

```
* * * * * * * * * * * * * * * * * * * * * * *Library.prg
* - - Sample Program to print database with memo field
USE Lib2
GO TOP
```

```
DO WHILE .NOT. EOF()
    ? "AUTHOR      :", Author
    ? "Title       :", Title
    ? "Publisher   :", Pub
    ? "Date        :", Date
    ? "Keywords    :", Topics
    ?
    ? Abstract
    ?
    ?
    SKIP
ENDDO(while not eof)
```

وعند تنفيذ هذا البرنامج تظهر البيانات الواضحة من الشكل (٢ - ٥) .

وهناك قصور واحد فى استخدام حقول الملاحظات ، وهو أنه لايمكن البحث عن الملاحظات التى تحتوى على كلمة معينة مثلا أو موضوع معين . فمثلا عند استخدام الأمر التالى :

```
LIST FOR "Computer" $ Abstract
```

للبحث عن السجلات التى تحتوى على كلمة (Computer) فى حقل الملاحظات (Abstract) . فى هذه الحالة يلاحظ ظهور الرسالة التالية :

```
Operation with memo field invalid
```

ولعلاج هذه المشكلة يمكن استخدام حقل آخر للبحث يسمى (Keywords) كما فى المثال السابق ، حيث يتم وضع بعض رؤوس المواضيع (Topics) التى يمكن بعد ذلك البحث من خلالها .

فمثلا عند كتابة الأمر التالى :

```
LIST FOR "Computer" $ Keywords
```

يتم عرض بيانات جميع السجلات التى تحتوى على كلمة (Computer) فى حقل الملاحظات .

Record no. : 1
Author : Garton , J ,
Title : Decission Support Systems
Publisher : Byte Magazine
Date : 03/01/89
Key words : Support,knowledge Maker , Mind games

This article discusses automated Decission
Support sytems used in modern business.
Mini and micro--------------------------

Record no. : 2
Author : Franklin B. W.
Title : Automated MBO
Publisher : MBO Monthly
Date : 05/01/89
Key words : Management , MBO , Microcomputer

Describes several automated systems ------

شكل (٢ - ٥)



الفصل الثالث

البحث السريع



هذا الفصل يركز على الوسائل المختلفة (Techniques) التي يمكن استخدامها فى كتابة البرامج لزيادة سرعة التشغيل بدرجة كبيرة . ومن أهم العمليات التي تؤثر بدرجة كبيرة فى سرعة تنفيذ البرنامج عملية البحث عن بيان معين خلال ملف قاعدة البيانات ، حيث أن ذلك قد يستغرق أياما فى الملفات الكبيرة اذا لم يتم كتابة البرامج بالصورة السليمة . أما عند استخدام الوسائل المختلفة التى سيتم شرحها فى هذا الباب فقد يصل زمن البحث الى دقائق معدودة وربما ثوان مهما كبر حجم ملف قاعدة البيانات . وأول هذه الوسائل هو استخدام الفهرس (Index) فى ترتيب سجلات الملف .

٣ - ١ استخدام الفهرس

أقرب وأوضح مثال لتأثير الفهرس على سرعة البحث عن البيانات هو استخدام فهرس الكتاب . نفرض مثلا أننا نقرأ فى كتاب عن الحاسبات ونريد أن نبحث عن موضوع نظم الخبرة ، فهناك طريقتان للبحث ، الأولى عن طريق فرز صفحات الكتاب صفحة صفحة حتى نصل الى الصفحة التى تحتوى على هذا الموضوع ، واذا كان الكتاب كبيرا فان البحث قد يستغرق مدة طويلة .والطريقة الثانية هي الذهاب مباشرة الى الفهرس الموجود فى آخر الكتاب والبحث فى الفهرس عن كلمة(نظم الخبرة)وتحديد رقم الصفحة المناظر ثم الذهاب الى هذه الصفحة.وعملية البحث فى فهرس الكتاب لن تأخذ وقتا كبيرا لأن هذا الفهرس يكون مرتبا بالترتيب الهجائى .

وما يحدث مع برامج عائلة (DBase) هو نفس الشيء تقريبا ، حيث تكون هناك طريقتان للبحث عن بيان معين فى ملف قاعدة البيانات . الطريقة الأولى عن طريق قراءة كل سجل من سجلات الملف للوصول الى السجل الذى يحتوى على البيان المطلوب البحث عنه . والطريقة الثانية هى انشاء ملف فهرس (Index) بناء على الحقل الذي يحتوى على البيان المطلوب البحث عنه ، مثل الاسم ، والبحث عن البيان المطلوب خلال هذا الفهرس ، وتحديد رقم السجل المقابل له ثم الذهاب الى هذا السجل .

وفى برامج عائلة (DBase) يتم انشاء الفهرس باستخدام الأمر (INDEX ON) . كما يتم تشغيل هذا الفهرس بكتابة اسمه عند فتح ملف قاعدة البيانات كالآتى مثلا :

```
USE Cadets INDEX Name
```

كما يتم استخدام الأمر (FIND) أو الأمر (SEEK) فى البحث عن البيان المطلوب .

وعند اجراء أى عمليات على سجلات ملف قاعدة البيانات مثل اضافة سجلات جديدة أو مسح سجلات أو تعديل بيانات سجل معين ، يجب مراعاة فتح ملف الفهرس أولا قبل اجراء هذه العمليات ، حيث أن فتح ملف الفهرس يؤدى الى ادخال أى تعديل يتم على ملف قاعدة البيانات على هذا الفهرس .

٣ - ٢ طرق البحث

بالاضافة الى السرعة التي يوفرها استخدام الفهرس في ترتيب السجلات فان هناك طرقا مختلفة للبحث عن السجلات توفرها برامج عائلة (DBase) وتتفاوت سرعة البحث من خلالها .لذلك فمن المهم عرض طرق البحث المختلفة ومقارنة سرعة البحث في كل طريقة حتى يختار مخطط البرامج الطريقة المناسبة التي تزيد من كفاءة وسرعة تنفيذ البرنامج .

فمثلا نفرض أن هناك ملف قاعدة بيانات اسمه (Test.dbf) يحتوى على ألف سجل .ونفرض أن هناك عشرة سجلات تحتوى على الاسم (Mohamed) مثلا . ويراد عرض بيانات هذه السجلات على الشاشة . فى هذه الحالة نقوم بمقارنة طريقتين مختلفتين لتنفيذ المطلوب .

فى الطريقة الأولى يتم استخدام الأمر(LIST) مع كلمة (FOR) لادخال شرط البحث . والسطور التالية توضح ذلك :

```
CLEAR
USE Test INDEX Name
ACCEPT "List what name ? " TO Search
LIST FOR Name = Search
```

وعند تنفيذ هذا البرنامج يتم مسح الشاشة ويظهر السؤال التالى :

```
List what name ?
```

وعند ادخال الاسم (Mohamed) مثلا يتم تخزينه فى متغير الذاكرة الحرفى (Search) ، وتظهر بيانات السجلات العشرة التي تحتوى على هذا الاسم . والوقت الذى يستهلك فى عرض هذه السجلات يزيد عن الدقيقتين بقليل . وذلك في حالة استخدام الأقراص المرنة (Floppy Disks) . كما يستهلك حوالى ٢٢ ثانية عند استخدام القرص الصلب (Hard Disk) .

والطريقة الثانية لتنفيذ نفس هذه العملية هي استخدام الأمر (FIND) أو الأمر (SEEK) في تحديد رقم أول سجل يحتوى على الاسم (Mohamed) ثم استخدام (WHILE) لعرض باقي السجلات . والسطور التالية توضح ذلك :

```
CLEAR
USE Test INDEX Name
ACCEPT "List what name? " TO Search
SEEK Search
LIST WHILE Name = Search
```

وفي هذه الحالة يتم عرض بيانات السجلات العشرة فى خمس ثوان فى حالة استخدام الأقراص المرنة (Floppy Disks) ويتم ذلك فى أربع ثوان فى حاة استخدام القرص الصلب (Hard Disk).

٢ - ٣ البحث عن مدى معين

عند البحث عن مدى معين من السجلات محصور بين قيمة معينة لأحد الحقول وقيمة أخرى لهذا الحقل ، فان ذلك يمكن أن يتم باستخدام الأمر (LIST) مع كلمة (FOR). كما يمكن استخدام الفهرس مع استخدام الأمر (SEEK) و (WHILE) لتنفيذ نفس العملية . ولتوضيح الفرق بين الحالتين سنقوم بدراسة المثال التالي:

نفرض أنه يراد البحث عن السجلات التى تبدأ من تاريخ معين فى حقل التاريخ (Date) وتنتهي بتاريخ آخر . فى هذه الحالة نبدأ باستخدام الأمر (LIST) مع كلمة (FOR) كالآتى :

```
USE Test
CLEAR
STORE "    " TO Start , Finish
@ 10,2 SAY "Enter start date" GET Start;
     PICT "99/99/99"
@ 12,2 SAY "Enter ending date" GET Finish;
     PICT "99/99/99"
READ
STORE CTOD(Start) TO Start
STORE CTOD(Finish) TO Finish
LIST FOR Date >= Start .AND. Date <= Finish
```

في هذه الحالة يتم عرض بيانات السجلات المحصورة بين التاريخين الذين يكتبهما المستخدم .

وعند استخدام الطريقة الأخرى يتم كتابة السطور التالية :

```
USE Test INDEX Dates
CLEAR
STORE "     " TO Start , Finish
@ 10,2 SAY "Enter start date";
  GET Start PICT "99/99/99"
@ 12,2 SAY "Enter ending date";
  GET Finish PICT "99/99/99"
READ
STORE CTOD(Start) TO Start
STORE CTOD(Finish) TO Finish
SEEK Start
LIST WHILE Date <= Finish
```

وفى هذه الحالة يتم عرض السجلات أسرع كثيرا من الطريقة الأولى، حيث أن البرنامج يبحث أولا عن السجل المحتوى على تاريخ البداية (Start) . وحيث أن الملف مفهرس على حقل التاريخ (Dates) ، فان البحث يتم عن السجلات التى تبدأ من السجل الذى تم تحديده بواسطة الأمر (SEEK) وتنتهى بالتاريخ الموجود فى المتغير (Finish) . ومع أن هذه الطريقة أسرع كثيرا من الطريقة الأولى الا أنها تنطوى على شيء من الخطورة . حيث أن الأمر (SEEK) اذا لم يجد السجل المحتوى على تاريخ البداية (Start) فان البرنامج لايعرض أى سجلات . لأن مؤشر السجلات (Record Pointer) ينتقل الى آخر الملف (End of File) .
ولذلك يفضل فى هذه الحالة استخدام طريقة أخرى تجمع بين الطريقتين للاحتفاظ بسرعة تنفيذ البرنامج . ويتم توضيح ذلك من السطور التالية :

```
USE Test INDEX Dates
CLEAR
STORE "     " TO Start , Finish
@ 10,2 SAY "Enter start date";
  GET Start PICT "99/99/99"
@ 12,2 SAY "Enter ending date";
  GET Finish PICT "99/99/99"
READ
STORE CTOD(Start) TO Start
```

```
STORE CTOD(Finish) TO Finish
SEEK Start
IF FOUND()
   LIST WHILE Date <= Finish
ELSE
   LIST FOR Date >= Satart .AND. Date <= Finish
ENDIF(not found)
```

وهناك طريقة أخرى لتنفيذ نفس المطلوب عن طريق استخدام الأمر (SET FILTER TO) وذلك كالآتي مثلا :

```
SET FILTER TO DATE >= Start .AND. DATE <= Finish
LIST
```

٣- ٤ التنفيذ السريع للعمليات الحسابية

هناك بعض الأوامر الحسابية مثل (COUNT) ، (SUM) ، (AVERAGE) . وهذه الأوامر يمكن أيضا تقليل وقت تنفيذها للعملية الحسابية بدرجة كبيرة . فمثلا لحساب عدد السجلات التى تحتوى على الاسم (Mahmoud) يمكن كتابة السطرين التاليين :

```
USE Test
COUNT FOR Name = "Mahmoud"
```

وهذه الطريقة تستهلك حوالى (١٥) ثانية للوصول الى النتيجة المطلوبة ، وهى وجود عشرة سجلات تحتوى على هذا الاسم .

ويمكن تقليل هذا الوقت بدرجة كبيرة جدا بكتابة الأوامر التالية :

```
USE Test INDEX Name
FIND Mahmoud
COUNT WHILE Name = "Mahmoud"
```

فى هذه الحالة يتم تنفيذ المطلوب فى حوالى ثانية واحدة .

ونفس هذه الطريقة يمكن استخدامها مع الأمر (SUM) والأمر (AVERAGE) .

٣ - ٥ زيادة سرعة طباعة التقارير

يمكن استخدام طريقتين أيضا في طباعة التقارير . ولتوضيح الفرق بينهما ، يتم كتابة السطرين التاليين الذين يمثلان الطريقة الأولى .

```
USE Test INDEX Name
REPORT FORM Rep1 FOR Name ="Mahmoud"
```

وهذه الطريقة تستغرق حوالى ٣٠ ثانية فى عرض التقرير على الشاشة . والسطور التالية توضح الطريقة الثانية :

```
USE Test INDEX Name
FIND Mahmoud
REPORT FORM Rep1 WHILE Name = "Mahmoud"
```

وهذه الطريقة تستغرق حوالى ٦ ثوان فى عرض التقرير على الشاشة .

٣ - ٦ النسخ السريع للسجلات

عندما يراد نسخ مجموعة من السجلات فى ملف مؤقت (Temporary) ، يمكن تنفيذ ذلك بطريقتين :

الطريقة الأولى يتم توضيحها من السطرين التاليين :

```
USE Test INDEX Name
COPY TO Temp FOR Name = "Mahmoud"
```

وهذه العملية تستهلك حوالى ٢٠ ثانية على القرص الصلب (Hard Disk) .

والطريقة الثانية يتم توضحيها من السطور التالية :

```
USE Test INDEX Name
FIND Mahmoud
COPY TO Temp WHILE Name = "Mahmoud"
```

هذه الطريقة تؤدى الى تقليل زمن التنفيذ الى مايقرب من ثانيتين .

٣ - ٧ التعامل مع ملفات فهرس متعددة

يحتاج مخطط البرامج الى ترتيب السجلات ترتيبا مختلفا حسب العملية المطلوب اجراؤهـــا ، فى هذه الحالة يمكنه استخدام عدة ملفات فهرس . وبرامج عائلة (DBase) تتيح لمخطط البرامج انشاء أى عدد من السجلات ، ولكنه لايستطيع فتح أكثر من سبعة ملفات فهرس فى نفس الوقت .

فمثلا فى برنامج شئون الطلبة (Cadets) يمكن انشاء فهرس بناء على حقل الاسم (Name) كالآتى :

```
USE Cadets
INDEX ON Name TO Name
```

كما يمكن انشاء فهرس بناء على رقم الفرقة مثلا (Class) كالآتى :

```
USE Cadets
INDEX ON CLass TO Class
```

ويمكن فتح الفهرسين معا بكتابة السطر التالى :

```
USE Cadets INDEX Name , Class
```

وترتيب كتابة ملفات الفهرس مهم جدا فى هذه الحالة ، حيث أن الملف الأول (Name) يصبح الفهرس الرئيسى (Primary) . فعند استخدام أى أمر من أوامر التعامل مع السجلات مثل (LIST) ، (REPORT) ، فان البرنامج يعرض السجلات مرتبة حسب الفهرس الرئيسى . واذا تساوت بيانات بعض السجلات فى حقل الفهرس الرئيسى يتم ترتيبها بناء على الفهرس الثانى وهكذا . كما أن الأوامر (SEEK) ، (FIND) تبحث خلال الفهرس الرئيسى فقط .

واذا أريد تعديل ترتيب ملفات الفهرس يمكن كتابة السطر التالى مثلا :

```
USE Cadets INDEX Class , Name
```

فى هذه الحالة يصبح الملف الأول (Class.ndx) هو الفهرس الرئيسى . ويراعى دائما فتح جميع ملفات الفهرس التى سبق انشاؤها ثم تعديل ترتيبها بعد ذلك حسب الحاجة . وذلك لأن أى تعديل يحدث فى السجلات مثل اضافة سجلات جديدة أو مسح سجلات أو تعديل سجلات يؤدى الى تحديث ملفات الفهرس المفتوحة . أما ملفات الفهرس غير المفتوحة فلا يتم تحديثها وبالتالى تصبح غير مطابقة للوضع الحالى للسجلات فى ملف قاعدة البيانات . ولعلاج ذلك يتم اعادة انشاء الفهرس من جديد . ويتم ذلك عن طريق فتح ملفات الفهرس التى سبق انشاؤها ثم استخدام الأمر (REINDEX) وذلك كالآتى :

```
USE Cadets INDEX Name , Class
REINDEX
```

الفصل الرابع

خطوات تصميم النظام

عادة يبدأ تصميم النظام بمجرد فكرة . ثم تنمو هذه الفكرة تدريجيا حتى تنتج النظام الكامل . وعادة يبدأ مخطط البرنامج بأن يسأل نفسه (من أين أبدأ ؟) . وعندما يجيب على هذا السؤال ويبدأ فى التنفيذ ، فانه يسأل نفسه بعد كل خطوة (أين أذهب بعد هذه الخطوة) . ومن مجموع هذه الخطوات يصل فى النهاية الى التصميم النهائى للنظام .

وخطوات التصميم بصفة عامة يمكن شرحها كالآتى :

١ - تعريف المشكلة أو الهدف من النظام .
٢- توصيف المدخلات والمخرجات .
٣ - تصميم هيكل قاعدة البيانات (Database Structure) .
٤- تقسيم البرنامج الى برامج فرعية (Modules) يؤدى كل منها وظيفة محددة .
٥ - كتابة البرامج الفرعية .
٦- اختبار وتصحيح البرنامج .

وسوف يتم شرح كل خطوة من هذه الخطوات فى هذا الباب .

٤ - ١ تعريف المشكلة (Problem Definition)

أول خطوة فى تصميم النظام هى تحديد المشكلة المطلوب حلها أو الهدف العام للنظام . وكلما كان هذا الهدف محددا وواضحا كان تنفيذ الخطوات التالية أسهل . وأول خطوة فى هذا التحديد تبدأ من اسم النظام نفسه مثل (نظام معلومات شئون الطلبة) . وهذا يعني أن المطلوب انشاء نظام يتيح الحصول على معلومات معينة عن الطلبة . ولكن هذا الاسم وحده يكون غامضا بعض الشيء . لذلك يتم توضيحه قليلا عن طريق معرفة خصائص المستخدم النهائي لهذا النظام . فاذا كان المطلوب انشاء نظام معلومات يتيح للمستخدم العادى الذى ليس له أى خبرة بالحاسب استخدام النظام واسترجاع المعلومات المطلوبة ، فى هذه الحالة يصبح من السهل على مخطط البرامج تحليل هذا الهدف الى خطوات محددة مثل الآتى :

١- اضافة بيانات طلبة جدد .
٢- طباعة تقارير متضمنة بيانات طالب معين أو مجموعة من الطلبة .
٣ - تعديل بيانات أى طالب .
٤- مسح بيانات أى طالب .

كما يمكن تحليل هذه الخطوات الى خطوات أخرى أكثر تحديدا .

٤ - ٢ توصيف المدخلات والمخرجات (Input/Output)

الخطوة الثانية فى تصميم النظام هى تحديد ما يجب ادخاله الى الحاسب (Input) وما يجب أن نحصل عليه منه (Output) . ولايهم في هذه المرحلة كيفية الادخال أو الاخراج . فمثلا فى نظام معلومات شئون الطلبة (Cadets) نريد الحصول على الآتى من الحاسب .

١ - تقارير بيانات الطلبة (Reports) تحتوى على الاسم والعنوان ورقم التليفون ... الخ .
٢ - تقارير مختصرة للطلبة (Labels) تحتوى على الاسم والعنوان .

وعادة يتم تحديد المدخلات عن طريق هذه المخرجات . لذلك نستطيع تحديد المدخلات الآتية :

```
Name
Nationality
Address
Phone number
----
----
```

٤ - ٣ تصميم قاعدة البيانات

الخطوة الثالثة هى تصميم قاعدة البيانات المطلوبة . وفى هذه المرحلة يجب تحديد نوع كل حقل من حقول ملف قاعدة البيانات . وهنا يجب التمييز بين الحقول العددية (Numeric) وبين الحقول الحرفية (Character) التى تحتوى على أعداد . حيث أن المقصود بالحقول العددية عادة هو الأعداد التى لايتم استخدامها فى عمليات حسابية فمثلا رقم التليفون يتم ادخاله كحقل حرفى وليس عدديا .

ويجب فى هذه المرحلة أيضا انشاء ملف الفهرس . ويجب تحديد نوع هذا الملف حسب الحقل المطلوب البحث بناء عليه . فمثلا فى نظام معلومات شئون الطلبة يتم كتابة السطرين التاليين لانشاء ملف الفهرس :

```
USE Cadets
INDEX ON Name TO Name
```

٤ - ٤ التصميم المجزأ للنظام (Modular Design)

كما سبق الايضاح ، فان أسهل طريقة لتصميم النظام هى تقسيمه الى برامج صغيرة (Modules) كل منها يحقق وظيفة محددة .

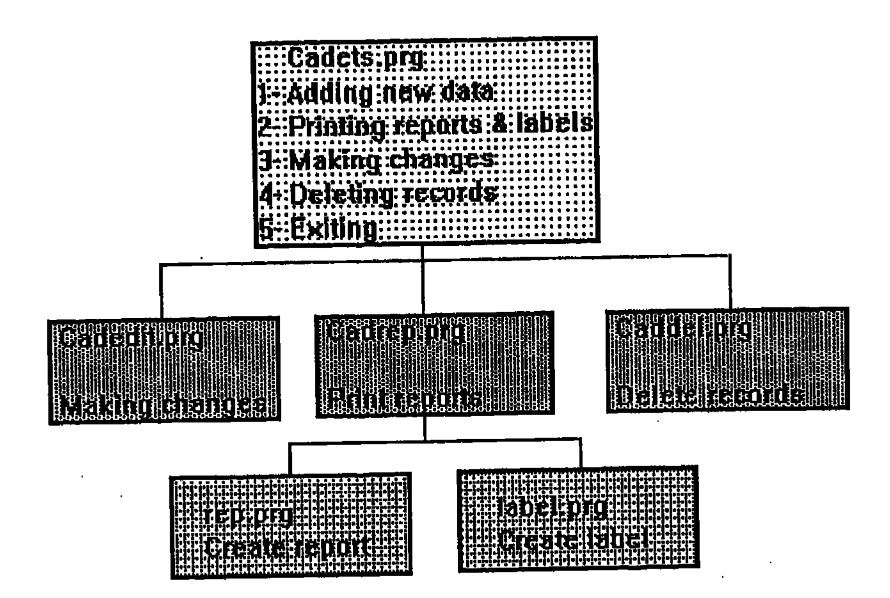
ويستخدم الشكل الهرمى (Hierarchical) فى تحديد البرنامج الرئيسى والبرامج الفرعية المتفرعة منه . فمثلا فى نظام معلومات شئون الطلبة (Cadets)، يكون البرنامج الرئيسى هو البرنامج الذى يقوم بعرض القائمة الرئيسية (Main menu) . والبرامج الفرعية هى التى تحقق كل اختيار من اختيارات القائمة . والبرنامج الرئيسى مثلا يعرض القائمة التالية :

```
-----------------------------------------------
        Cadets Information System

1 - Add new names and addresses
2 - Print reports
3 - Make changes
4 - Delete names and addresses
5 - Exit

Enter choice:
-----------------------------------------------
```

والشكل التالى يوضح التركيب الهرمى للبرنامج :


Cadets.prg
1- Adding new data
2- Printing reports & labels
3- Making changes
4- Deleting records
5- Exiting
Cadedit.prg
Making changes
Cadrep.prg
Print reports
Caddel.prg
Delete records
rep.prg
Create report
label.prg
Create label

الفصل الخامس

كتابة البرامج



يتم كتابة البرامج من خلال برنامج (DBase III+) أو برامج عائلة (DBase)الأخرى عن طريق كتابة ملفات الأوامر (Command Files) التى تحتوى على الأوامر المطلوب تنفيذها بالتسلسل المنطقى المطلوب .

ويستخدم الأمر (MODIFY COMMAND) فى كتابة ملفات الأوامر . حيث أن هذا الأمر يؤدى الى تشغيل المصحح الخطى (Text Editor) الخاص ببرنامج (DBase III+) الذى يتم عن طريقه كتابة ملفات الأوامر وتصحيحها . كما يستخدم الأمر (DO) فى تشغيل ملفات الأوامر .

٥ - ١ انشاء ملفات الأوامر

كما سبق الايضاح فان لبرنامج (DBase III+) المصحح الخطى الخاص به . وهذا المصحح الخطى يتم تشغيله عن طريق الأمر (MODIFY COMMAND) . فمثلا لانشاء الملف (Test) يتم كتابة الأمر التالى :

```
MODIFY COMMAND Test
```

وعند الضغط على مفتاح الادخال ، تظهر شاشة خالية لكتابة الأوامر خلالها . وتظهر قائمة مساعدة (Help) توضح المفاتيح التى يتم عن طريقها التحكم فى الكتابة .وهذه المفاتيح يتم شرح وظائفها من خلال الجدول الموضح بالشكل (٥ - ١) .

ويمكن كتابة سطرين من الملف (Test) للتمرين على كتابة ملفات الأوامر كالآتى مثلا :

```
CLEAR
? "Good morning"
```

ويتم الضغط على مفتاح الادخال بعد كتابة كل سطر . واذا حدث خطأ فى الكتابة يتم استخدام مفاتيح التصحيح الموجودة فى الجدول فى الوصول الى الحروف المطلوب تعديلها . ثم يتم تخزين الملف عن طريق كتابة (^End) أو كتابة (^W) . وهذا يؤدى الى تخزين الملف (Test) والعودة الى مشيرة النقطة .

المفتاح			الوظيفة
↑	أو	E	تحريك المؤشر سطرا لأعلى
↓	أو	X	تحريك المؤشر سطرا لأسفل
S	أو	<-	تحريك المؤشر حرفا لليسار
^D	أو	->	تحريك المؤشر حرفا لليمين
Del	أو	G	مسح الحرف فوق المؤشر
Ins	أو	^V	التحويل الى حالة الاضافة (Ins) أو الكتابة الفوقية
End	أو	^F	تحريك المؤشر كلمة الى اليمين
Home	أو	^A	تحريك المؤشر كلمة الى اليسار
^N			اضافة سطر خال مكان المؤشر
^T			مسح كلمة يمين المؤشر
^Y			مسح سطر مكان المؤشر
^End	أو	^W	تخزين ملف الأوامر
Esc	أو	^Q	الرجوع الى مشيرة النقطة دون تخزين الملف
^Ka			يعيد تشكيل النص (يستخدم عادة مع حقول الملاحظات)
^KF			البحث عن كلمة معينة فى النص
^KL			تحديد مكان الكلمة الثانية التي يتم البحث عنها
^KR			قراءة ملف خارجي فى مكان المؤشر
^KW			كتابة ملف معين فى ملف آخر باسم آخر

شكل (٥ - ١)

والبرنامج يضيف الامتداد (prg) لاسم الملف . وعندما يراد تشغيل هذا الملف يتم كتابة الآتى :

```
DO Test
```

وعند الضغط على مفتاح الادخال يلاحظ مسح الشاشة وظهور الآتى :

```
Good morning
```

ويلاحظ أن البرنامج (Test.prg) قام بتنفيذ السطرين ، حيث بدأ بمسح الشاشة (Clear) ثم عرض السطر السابق .

وعندما يراد تعديل هذا البرنامج مثلا يتم كتابة السطر التالى :

```
MODIFY COMMAND Test
```

وفى هذه الحالة تظهر السطور السابق كتابتها ويتم تعديلها .

٥ - ٢ التفاعل مع المستخدم

فى المثال السابق كان المطلوب فقط عرض رسالة على الشاشة . ولكن فى معظم البرامج يكون مطلوبا سؤال المستخدم وانتظار اجابته ، ثم تخزين هذه الاجابة فى متغير ذاكرة . وبناء على قيمة هذا المتغير يتم تنفيذ عملية معينة . وهذا يشبه الحديث بين شخصين وتصرف كل منها بناء على ذلك . وهناك عدة أوامر يتم استخدامها فى تحقيق هذا التفاعل (Interaction) بين المستخدم والحاسب . وهذه الأوامر يتم القاء الضوء عليها فى الأجزاء التالية .

٥ - ٢ - ١ الأمر (ACCEPT)

يستخدم هذا الأمر فى عرض رسالة للمستخدم وانتظار اجابته على هذه الرسالة ، ثم تخزين هذه الاجابة فى متغير ذاكرة حرفى . فمثلا يمكن كتابة السطر التالى :

```
ACCEPT "Send report to printer ? (Y/N)" TO Pr
```

وعند الضغط على مفتاح الادخال تظهر الرسالة التالية على الشاشة :

```
Send report to printer ? (Y/N)
```

وينتظر البرنامج حتى يدخل المستخدم الاجابة ، ثم يقوم بتخزين هذه الاجابة فى المتغير (Pr) . وهذا الأمر يفضل استخدامه عندما تكون الاجابة المنتظرة من المستخدم حرفية وليست عددية . وهذا لايعنى أنه لايقبل الاجابة العددية : ولكنه يعامل هذه الاجابة كقيمة حرفية حتى لو كانت عددا .

٥ - ٢ - ٢ الأمـــر (INPUT)

هـذا الأمـر يشبـه الأمـر (ACCEPT) حيـث يعـرض رسـالـة للمستخـدم وينتـظـر اجابته على هذه الرسالة . ولكن الأمر (INPUT) يتعامل مع اجابة المستخـدم حسب نوعها سواء كانت حرفية أو عددية . فاذا كانت الاجابة عـددية يـقـوم بانشاء متغير عددى . فمثلا يمكن كتابة السطر التالى :

```
INPUT "Enter your age" TO Age
```

وهذا يؤدى الى ظهور الرسالة التالية على الشاشة :

```
Enter your age
```

وينتظر البرنامج من المستخدم ادخال قيمة معينة . وعلى حسب نوع هذه القيمة يقوم بانشاء متغير ذاكرة من نفس النوع .

٥- ٢ - ٣ الأمـــر (WAIT)

يستخـدم هذا الأمر مثـل الأمرين السـابقين فى عرض رسالة للمستخدم ولكنـه ينتظـر من المستخدم الضغـط على أى مفتاح حتى يستمر تنفيـذ البـرنـامـج . ويستخدم عادة عندما يرا د ايقاف تنفيذ البرنامج ايقافا مؤقتا (Pause) حتى يستطيع المستخدم قراءة بيانات معينة على الشاشة . وهذا الأمر يمكن كتابتـه دون كتابة أى شىء بعده . وهـذا يؤدى الى ظهـور الرسالة التالية :

```
Press any key to continue
```

وهذه هى الرسالة المبدئية (Default) .

كما يمكن كتابة أى رسالة أخرى كالآتى مثلا :

```
WAIT "Press any key To return to main menu "
```

كما يمكن استخدامه دون عرض أى رسائل وذلك كالآتى :

```
WAIT "   "
```

كما يمكن استخدامه فى انشاء متغير ذاكرة (Pr) يتم فيه تخزين الحرف الذى يكتبه المستخدم وذلك كالآتى مثلا :

```
WAIT "Send report to printer ? (Y/N)" TO Pr
```

وهذا المتغير الذى يتم انشاؤه يكون متغيرا حرفيا . واذا أريد استخدام هذا الأمر فى استقبال قيمة عددية من المستخدم يتم كتابة الآتى :

```
WAIT "Enter your choice (1-5)" TO choice
Choice = VAL(Choice)
```

وفى هذه الحالة يتم انشاء متغير ذاكرة حرفى (Choice) يحتوى على الرقم الذى يدخله المستخدم ثم يتم تحويل هذا المتغير الى قيمة عددية .

٥- ٢ - ٤ الأمـــر (@...SAY...GET)

يستخدم هذا الأمر مثل الأوامر السابقة فى عرض رسالة للمستخدم وانتظار الاجابة على هذه الرسالة ، ثم تخزين هذه الاجـابة فى متغيـــر ذاكرة . ولكـن هذا الأمر يمتاز بالقدرة على التحكم فى مكان ظهور الرسالة على الشاشة عن طريق الاحداثيات التى يتم كتابتها بعد الحرف (@) .

وهذا الأمر يختلف عن الأوامر السابقة فى أنه يلزم قبل استخدامه انشاء متغير الذاكرة أولا . كما يستخدم الأمر (READ) بعد ذلك فى تخزين اجابة المستخدم فى هذا المتغير الذى سبق انشاؤه . فمثلا عندما يراد سؤال المستخدم عن الاختيار المطلوب من الشاشة وتخزين هذا الاختيار فى المتغير(Choice) يتم كتابة السطور التالية :

```
Choice = 0
@ 10,5 SAY "Enter choice" GET Choice
READ
```

وهذه الأوامر تبدأ بانشاء المتغير العددى(Choice) ، ثم عرض الرسالة (Enter Choice) فى السطر العاشر والعمود الخامس . ثم ينتظر البرنامج من المستخدم ادخال أى قيمة عددية ، ويقوم بتخزينها فى المتغير (Choice).

٥ - ٣ الحلقة التكرارية

تعد الحلقة التكرارية أحد الأشكال الشائعة الاستخدام فى البرامج بصفة عامة وفي برامج عائلة (DBase) بصفة خاصة . ويتم تكوينها في برامج عائلة (DBase) بواسطة الأمر (DO WHILE) . وهي تبدأ عادة بهذا الأمر وتنتهى بالأمر (ENDDO) . ولتوضيح وظيفة الحلقة التكرارية يتم كتابة برنامج أسمه (Count) مثلا . هذا البرنامج يتكون من السطور التالية :

```
CLEAR
SET TALK OFF
STORE 1 TO X
DO WHILE X <= 20
? X
X = X + 1
ENDDO
```

وعند تنفيذ هذا البرنامج يظهر الآتى على الشاشة :

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

ويمكن توضيح تنفيذ البرنامج لهذه الحلقة التكرارية كالآتى :

يبدأ البرنامج بالأمر (CLEAR) الذى يؤدى الى مسح الشاشة . ويلى ذلك الأمر (SET TALK OFF) الذى يمنع رسائل البرنامج من الظهور على الشاشة أثناء تنفيذه . ثم يتم انشاء المتغير (X) بواسطة الأمر (STORE) واعطاؤه القيمة (1) . ثم تبدأ الحلقة التكرارية بالأمر (DO WHILE X <= 20) . وهذا الأمر يعنى البدء فى تنفيذ الحقة التكرارية التى يستمر تنفيذها طالما كانت قيمة (X) أصغر من أو تساوى (٢٠) . والأمر (? X) يؤدى الى عرض قيمة (X) على الشاشة . كما يستخدم الأمر (X = X + 1) فى زيادة قيمة المتغير (X) بواحد فى كل مرة يتم فيها تنفيذ الحلقة التكرارية . والأمر (ENDDO) يحدد نهاية الحلقة التكرارية .

والأمر (DO WHILE) يستخدم عادة عندما يراد المرور على سجلات ملف قاعدة البيانات وتنفيذ عمليات معينة عليها. وفى هذه الحالة فان هذا الأمر يتم كتابته عادة كالآتى :

```
DO WHILE .NOT. EOF()
```

وهذا يعنى تنفيذ الحلقة التكرارية طالما لم يتم الوصول الى نهايـة الملف (EOF()) . وهى اختصار (END OF FILE) .

٥- ٤ اتخاذ القرار بواسطة الأمر (IF)

يستخدم هذا الأمر عندما يراد اتخاذ قرار بناء على معلومات معينة . ويمكن توضيح هذا الأمر عن طريق كتابة برنامج نسميه مثلا (IFTest) ويتم كتابته كالآتى :

```
MODIFY COMMAND IFTest
```

وفى هذه الحالة تظهر شاشة الكتابة ويتم كتابة السطور التالية :

```
********************************** IFTest.prg
* - - - - - Program to test the (IF) Command
CLEAR
ACCEPT "Turn on printer ? (Y/N)" TO Pr
* - - - - - If answer is yes , then
IF UPPER(Pr) = "Y"
       SET PRINT ON
```

```
          ? "You chose the printer"
          EJECT
          SET PRINT OFF
* - - - - -  If answer is not yes , then
ELSE
          CLEAR
          ? "You chose the screen"
ENDIF
```

وأول سطرين من هذا البرنامج همـــا سطران مخصصان لملاحظات مخطط البـرامج . لذلك فانهما يبدآن بحرف النجمة (*) . وهذه الملاحظات تفيد فى تـذكير مخطط البرامج باسم البرنامج ووظيفته . كما أنها تفيد داخل البرنامج فى تحديد وظيفة كل مجموعة من الأوامر تؤدى وظيفة محددة . والأمر (CLEAR) يؤدى الى مسح الشاشة .

والسطر التالى يؤدى الى عرض السؤال ((Turn on Printer? (Y/N)) على الشاشة وانتظار اجابة المستخدم . ثم تخزين هذه الاجابة فى المتغير (Pr) .

والسطر التالى يقوم باختبار الحرف الذى يكتبه المستخدم فاذا كان هذا الحـرف بعد تحويله الى حرف كبير (Uppercase) مساويا للحرف (Y) ، يتم تنفيذ مجموعة الأوامر المحصورة بين الأمر (IF) والأمر (ELSE) . وفـائدة استخدام الـدالة (UPPER) هى الحصول على نفس النتيجة سواء أدخل المستخدم الحـرف كبيـرا (Uppercase) أو صغيرا (LowerCase) . وأذا أدخل المستخدم أى حرف آخر غير (Y) أو (y) . يقوم البرنامج بتخطى الأوامر المحصورة بين (IF) ، (ELSE) وينفذ الأوامر المحصورة بين (ELSE) ، (ENDIF) . ولاختبار هذا البرنامج يتم كتابة هذا الأمر من مشيرة النقطة :

```
DO IFTest
```

وفى هذه الحالة يتم مسح الشاشة ثم تظهر الرسالة التالية :

```
Turn on Printer ? (Y/N)
```

فاذا تم كتابة (Y) والضغط على مفتاح الادخال يتم طباعة الرسالة التالية على طابعة :

```
You chose the printer.
```

ثم يتم نقل صفحة على الطابعة (EJECT) .

واذا تم كتابة أى حرف آخر غير (Y) أو (y) يقوم البرنامج بمسح الشاشة وعرض الرسالة التالية :

```
You chose the screen.
```

وتستخدم كلمة (ENDIF) فى انهاء مجموعة الأوامر الخاصة بالأمر (IF) . ويجب ملاحظة أن كل أمر (IF) له كلمة (ENDIF) خاصة به . أما كلمة (ELSE) فهى اختيارية .

ويمكن استخدام الدالة (IIF) فى تحقيق نفس العمل الذى يؤديه الأمر (IF) مع توفير فى عدد السطور المطلوب كتابتها . فمثلا يمكن كتابة السطر التالى :

```
?IIF (X < 10, "Less Than" , "Greater Than")
```

هذا الأمر يختبر قيمة (X) فاذا كانت أصغر من (10) يتم عرض الرسالة (Less than) على الشاشة ، واذا كانت أكبر من (10) يتم عرض الرسالة (Greater than) على الشاشة .

ولتنفيذ نفس هذه العملية بواسطة الأمر (IF) يتم كتابة السطور التالية :

```
IF X < 10
        ? "Less Than"
ELSE
        ? "Greater Than"
ENDIF
```

ومن ذلك يلاحظ أن الدالة (IIF) قد وفرت فى عدد السطور .

٥ - ٥ اتخاذ القرار بواسطة الأمر (DO CASE)

الأمر (DO CASE) يعتبر صورة أكثر شمولا من الأمر (IF) ، حيث أنه يتيح للبرنامج الاختيار بين عدة حالات تبعا لقيمة متغير معين . ولتوضيح ذلك يمكن كتابة برنامج نسميه مثلا (CaseTest) . ولتنفيذ ذلك يتم كتابة الأمر التالى من مشيرة النقطة (Dot Prompt) .

```
MODIFY COMMAND CaseTest
```

ثم يتم كتابة الأوامر الموضحة بالشكل (٥ - ٢) . وعند الانتهاء يتم تخزين هـذا البرنامج بكتابة (^End) .

```
* * * * * * * * * * * * * * * * CaseTest.prg
* - - - - -  Test the DO CASE command
CLEAR
INPUT "Enter a number from 1 to 4" TO X
DO CASE
    CASE X = 1
       ? "You entered one"
    CASE X = 2
       ? "You entered two"
    CASE X = 3
       ? "You entered three"
    CASE X = 4
       ? "You entered four"
    OTHERWISE
       ? "Invalid number"
ENDCASE
```

شكل (٥ - ٢)

والسطر الأول من البرنامج بعد سطرى الملاحظات يؤدى الى مسح الشاشة . والسطر التالى يؤدى الى عرض الرسالة التالية على الشاشة :

```
Enter a number from 1 to 4
```

ثم ينتظر البرنامج حتى يتم ادخال رقم والضغط على مفتاح الادخال . فى هـذه الحالة يتم تخزين هذا الرقم فى المتغير العددى (X) .

ثم يبدأ اتخاذ القرار بناء على قيمة (X) . فاذا كانت (X) تساوى (1) يعرض البرنامج الرسالة التالية :

```
You entered one
```

ثم يترك باقى الاختيارات .

واذا كانت (X) تساوى (2) يعرض البرنامج الرسالة التالية :

You entered two

ثم يترك باقى الاختيارات . وهكذا .

أما اذا تم ادخال رقم لايحقق أيا من هذه الاختيارات ، يتم تنفيذ الأمر التالى لكلمة (OTHERWISE) . وهو يؤدى الى ظهور الرسالة التالية :

Invalid number

٥ - ٦ الكتابة التركيبية للبرامج

من المهم جدا كتابة البرامج بالطريقة التركيبية (Structured) . حيث أن هذه الطريقة تسمح لمخطط البرامج بالرجوع الى البرنامج واختباره وتصحيحه بسهولة كما تسمح لأى مخطط برامج بقراءة البرنامج الذى تمت كتابته بواسطة شخص آخر والقيام بتعديله أو تطويره حسب الحاجة .

ولكتابة البرنامج بهذه الطريقة يجب تنفيذ الآتى :

١ - استخدام الملاحظات الواضحة التى تشرح وظيفة كل مجموعة من الأوامر تؤدى وظيفة محددة .
٢ - تحريك بدايات السطور (Indent) فى الحلقات التكرارية والسطور الخاصة بالأمر (IF) والأمر(DO CASE) حتى تظهر بداية ونهاية السطور بوضوح .
٣ - اختيار الأوامر والمتغيرات التى توضح الوظيفة التى يتم تنفيذها .

ويمكن توضيح فائدة هذه الطريقة عن طريق مقارنة برنامجين أحدهما مكتوب دون مراعاة الطريقة التركيبية (Structured) . أنظر الشكل (٥ - ٢) . والآخر يتم فيه مراعاة هذه الطريقة . انظر الشكل (٥ - ٤) .

وفى البرنامج الأول يلاحظ أن قواعد الكتابة التركيبية غير مطبقة . حيث يلاحظ عدم وجود ملاحظات كافية توضح وظيفة كل مجموعة من الأوامر . كما يلاحظ عدم تحريك السطور للداخل (Indentation) فى السطور التى تحتاج الى ذلك . وهذا يؤدى الى عدم وضوح بداية الحلقة التكرارية ونهايتها وكذلك بالنسبة للسطور التالية للأمر (IF) أو الأمر (DO CASE) . وعندما تزيد الحلقات التكرارية كما يحدث فى معظم البرامج تصبح عملية اختبار البرنامج وتصحيحه عملية معقدة جدا .

```
* * * * * * * * * * * * * * * * *  Library.prg
* - - - -  Example of an unstructured program
USE Library
DO WHILE .T.
CLEAR
@ 1,20 SAY "Library Management system"
@ 3,25 SAY "1. Add new records"
@ 4,25 SAY "2. Print Reports"
@ 5,25 SAY "3. Edit data"
@ 6,25 SAY "4. Exit"
STORE 0 TO Choice
@ 8,20 SAY "Enter choice(1-4) " GET Choice
READ IF Choice = 1
APPEND
ELSE
IF Choice = 2
REPORT FORM Library
ELSE
IF Choice = 3
EDIT
IF Choice = 4
RETURN
ENDIF
ENDIF
ENDIF
ENDIF
ENDDO
```

شكل (٥ - ٢)

كما يلاحظ استخـدام بعض الأوامر التي قد تعطي معنى مخالفا للواقع مثل الأمر (DO WHILE .T.) . وهذا يعني أن الحلقة التكرارية سيتم تنفيـذها الـى مالانهاية . والحقيقة أنها لاتنفذ الى مالانهاية ، لأن اختيار المستخدم للرقم (4) يؤدى الى توقف تنفيذها .

والبرنامج بهذه الصورة سوف يتم تنفيذه كما أنه سوف يؤدى الوظيفة المطلوبة . ولكنه لايعتبر برنامجا موثقا (Documented) يمكن الرجوع اليه وتعديله أو تطويره .

```
* * * * * * * * * * * * * * * * *  Library.prg
* - - - - -  Library system main menu.
USE Library
STORE 0 TO CHOICE

DO WHILE Choice # 4
    CLEAR "
    @ 1,20 SAY "Library Management system "
    @ 3,25 SAY "1. Add new records "
    @ 4,25 SAY "2. Print Reports "
    @ 5,25 SAY "3. Edit data "
    @ 6,25 SAY "4. Exit
    @ 8,20 SAY "Enter choice(1-4) " GET Choice
    READ
    * - -  Branch according to user's request.

    DO CASE
        CASE Choice = 1
          APPEND
        CASE Choice = 2
          REPORT FORM Library
        CASE Choice = 3
          EDIT
    ENDCASE
ENDDO(while choice # 4)

* - - - - - -  when choice = 4 exit.
RETURN
```

شكل (٥ - ٤)

أما البرنامج الثانى (الشكل (٥ - ٤)) فانه يؤدى نفس وظيفة البرنامج السابق ولكنه مكتوب بالطريقة التركيبية . حيث يلاحظ أن كل الملاحظات واضحة تماما ومكتوبة بأسلوب واضح . كما يلاحظ أن كل السطور داخل الحلقة التكرارية تم تحريكها للداخل قليلا (Indentation) مما يجعل من السهل تحديد بداية الحلقة التكرارية ونهايتها (يمكن تحريك الأصبع بدءا من الأمر (DO WHILE) رأسيا حتى يصل الى الأمر (ENDDO) الخاص به) . كما يمكن تحديد بداية الأمر (DO CASE) ونهاية السطور الخاصة به بنفس الطريقة . كما تم استبدال الأوامر التى كانت تسبب شيئا من الغموض بأوامر أخرى واضحة . حيث تم استخدام الأمر (DO WHILE Choice # 4) بدلا من الأمر (DO WHILE .T.) .

كما يلاحظ استخدام الملاحظات بجانب كلمة (ENDDO) لتوضح لمخطط البرامج أي حلقة تكرارية تتبع لها هذه الكلمة . وهذا يفيد بصفة خاصة عندما تزيد الحلقات التكرارية فى البرنامج . حيث أن أى ملاحظات تكتب بعد كلمة (ENDDO) أو كلمة (ENDIF) لايشعر بها البرنامج وتعامل مثل أى ملحوظة مكتوبة بعد الحرف (*) .

الفصل السادس

وسائل التصحيح (Debugging)



٦ - ١ مقدمة

يجب أن يكون واضحا أنه نادرا ما يوجد البرنامج الذى يكتب أول مرة بدون أخطاء . والأخطاء عموما قد تكون بسيطة مثل الهجاء الخاطىء للأوامر (Misspelling) . وهذه الأخطاء يسهل اكتشافها عند تشغيل البرنامج . وقد تكون أخطاء منطقية (Logical) وهى عادة تكون معقدة ويصعب اكتشافها . حيث أنها قد لاتؤدى الى ايقاف البرنامج ولكنها تعطى نتيجة غير النتيجة المتوقعة منه .

وبرامج عائلة(DBase) توفر كثيرا من أدوات التصحيح (Debugging Tools) التى تسهل على مخطط البرامج اكتشاف الأخطاء وتصحيحها .

فمثلا عندما يجد البرنامج خطأ يؤثر فى تنفيذ البرنامج فانه يعرض السطر الذى يحتوى على الخطأ والبرنامج الذى يحتوى على الخطأ مع الرسالة التحذيرية التالية :

```
Cancel , Ignore or Suspend? (C,I or S)
```

وهذه الاختيارات تعنى الآتى :

١- الاختيار (Cancel)

هذا الاختيار يؤدى الى انهاء تشغيل البرنامج والعودة الى مشيرة النقطة (Dot Prompt) . وفى هذه الحالة تختفى كل متغيرات الذاكرة الخاصة (Private) التى تم انشاؤها خلال البرنامج .

٢- الاختيار (Suspend)

هذا الاختيار يؤدى الى توقف البرنامج مؤقتا مع ظهور الرسالة (Do suspend) . وفى هذه الحالة تظل متغيرات الذاكرة الخاصة موجودة فى الذاكرة . كما يمكن استكمال البرنامج فى أى وقت عن طريق كتابة الأمر (RESUME) من مشيرة النقطة .

٣ - الاختيار (Ignore)

يؤدى هذا الاختيار الى تخطى السطر المحتوى على الخطأ ومحاولة تنفيذ باقى سطور البرنامج اذا لم تكن معتمدة على هذا السطر .

ويمكن ايقاف البرنامج فى أى لحظة أثناء تنفيذه بالضغط على مفتاح الهروب (Esc) . وفى هذه الحالة تظهر نفس الاختيارات الثلاثة السابق شرحها .

٦- ٢ عرض الذاكرة (Memory)

من الوسائل الفعالة فى اكتشاف أخطاء البرنامج ، عرض الذاكرة (Memory) أثناء تنفيذ البرنامج .

فعند حدوث خطأ معين مثلا يمكن كتابة الأمر التالى :

```
DISPLAY MEMORY
```

فى هذه الحالة يتم عرض حالة الذاكرة المؤقتة (RAM) من حيث متغيرات الذاكرة الموجودة بها وأسمائها ومحتوياتها وأنواعها . ويمكن فى هذه الحالة اكتشاف سبب الخطأ عن طريق مراجعة أسماء متغيرات الذاكرة وأنواعها .

وللحصول على محتويات الذاكرة مطبوعة على الورق يتم كتابة السطر التالى :

```
DISPLAY MEMORY TO PRINT
```

كما يمكن عرض هيكل ملف قاعدة البيانات المفتوح عن طريق كتابة السطر التالى :

```
DISPLAY STRUCTURE
```

فى هذه الحالة يمكن اختبار أسماء الحقول وأنواعها واكتشاف أى خطأ موجود . كما يمكن استخدام الأمر (DISPLAY STATUS) لعرض أسماء ملفات قواعد البيانات المفتوحة وكذلك ملفات الفهرس .

٦- ٣ عرض التاريخ (History)

المقصود بالتاريخ (History) هو ذاكرة خاصة يتم فيها تخزين آخر عشرين أمرا تم ادخالها من خلال مشيرة النقطة (Dot Prompt). وهذا يؤدى الى امكانية الرجوع الى أى أمر تم ادخاله من آخر عشرين أمرا ، وذلك بالضغط على مفتاح السهم لأعلى (↑) مثلا . وهذا يوفر على مخطط البرامج كتابة الأمر عدة مرات ، حيث يكفى الضغط على مفتاح السهم لأعلى (↑) عدة مرات حتى يظهر الأمر المطلوب ثم الضغط على مفتاح الادخال .

ولكن هذا لاينطبق على ملفات الأوامر (Command files) . حيث أن أوامر الملف لايتم تخزينها فى ذاكرة التاريخ (History) الا بعد استخدام الأمر (SET DOHISTORY ON) لتجهيز ذاكرة التاريخ لتخزين الأوامر التى يتم تنفيذها تباعا .

فمثلا يمكن كتابة هذا الأمر قبل تنفيذ البرنامج . فاذا توقف البرنامج نتيجة خطأ معين يمكن استخدام الأمر (SUSPEND) فى انهاء البرنامج مؤقتا ، ثم يتم كتابة الأمر (DISPLAY HISTORY) أو الأمر (LIST HISTORY).

فى هذه الحالة يتم عرض آخر عشرين أمرا تم تنفيذها فى البرنامج وقد يؤدى هذا الى اكتشاف مكان الخطأ .

ويمكن زيادة عدد الأوامر التى يتم ادخالها فى ذاكرة التاريخ الى أكثر من عشرين أمرا عن طريق كتابة السطر التالى مثلا :

```
SET HISTORY TO 50
```

ويجب ملاحظة أن الأمر (SET DOHISTORY ON) يؤدى الى ابطاء تنفيذ البرنامج بدرجة كبيرة ، ولذلك يراعى عند الانتهاء من عملية التصحيح (Debugging) اعادة الأمر الى الوضع المبدئى . وذلك بكتابة السطر التالى :

```
SET DOHISTORY OFF
```

٦- ٤ استخدام الأمر (SET TALK ON)

عادة يتم كتابة الأمر (SET TALK OFF) فى أى برنامج حتى لاتظهر

الرسائل التي توضح كل خطوة يتم تنفيذها . وهذه الرسائل تكون مفيدة جدا عند تصحيح البرنامج . ولذلك يفضل أثناء اختبار البرنامج وتصحيحه كتابة الأمر (SET TALK ON) قبل تشغيل البرنامج . كما يتم متابعة الرسائل التى تظهر على الشاشة عند تنفيذ كل خطوة . فعند حدوث خطأ معين فى البرنامج ، قد تؤدى هذه الرسائل الى توضيح سبب هذا الخطأ .

٦- ٥ استخدام الأمر (SET ECHO ON)

عند كتابة هذا الأمر قبل تنفيذ البرنامج ، فان هذا يؤدى الى عرض كل سطر على الشاشة قبل تنفيذه . ويجب اعادة الأمر الى الوضع المبدئى (Default) عند الانتهاء من عملية التصحيح وذلك بكتابة السطر التالى :

```
SET ECHO OFF
```

٦- ٦ استخدام الأمر (SET STEP ON)

عند استخدام الأمر (SET ECHO ON) فان الرسائل التى توضح خطوات التنفيذ تظهر على الشاشة أثناء تنفيذ البرنامج كما سبق الايضاح . ولكن الرسائل فى هذه الحالة تظهر سريعة مع سرعة تنفيذ هذه الخطوات . فاذا أريد ابطاء ظهور هذه الرسائل حتى يستطيع مخطط البرامج متابعتها فيمكنه فى هذه الحالة استخدام الأمر (SET STEP ON) . وهذا يؤدى الى عرض كل أمر عند تنفيذه ، ثم توقف البرنامج مؤقتا ، ثم عرض الرسالة التالية :

```
Press SPACE to stop,S to suspend,or Esc to cancel
```

ويمكن فى هذه الحالة لمخطط البرامج الضغط على مسطرة المسافات (Space) لتنفيذ الأمر التالى . أو كتابة (S) لتعليق تنفيذ البرنامج ، وذلك عند ملاحظة خطأ معين مثلا ، أو الضغط على مفتاح الهروب (Esc) للخروج من البرنامج . وهذا يتيح لمخطط البرامج التحكم فى ظهور رسائل تنفيذ الأوامر أثناء تشغيل البرنامج مما يساعده على اكتشاف الأخطاء المنطقية (Logical Errors) فى البرنامج .

٦- ٧ استخدام الأمر (SET DEBUG ON)

يستخدم هذا الأمر لارسال خطوات تنفيذ البرنامج الى الطابعة . وهو يستخدم

مع الأمر (SET ECHO ON) والأمر (SET STEP ON) لمتابعة خطوات البرنامج خطوة خطوة حتى يتم اكتشاف مكان الخطأ .

٦ - ٨ أهم أخطاء كتابة البرامج

فى هذا الجزء يتم توضيح أهم رسائل الأخطاء (Error Messages) التى تظهر أثناء تنفيذ البرنامج ، ووسائل علاج هذه الأخطاء .

٦ - ٨ - ١ الرسالة (Data type mismatch)

تظهر هذه الرسالة عادة عند معاملة نوع معين من البيانات كنوع آخر . مثل معاملة البيانات الحرفية كعددية مثلا ، أو البيانات التاريخية كبيانات حرفية وهكذا . فمثلا عند كتابة السطر التالى :

```
LIST FOR Date = "01/30/90"
```

تظهر الرسالة (Data type mismatch). وذلك لأن الحقل (Date) حقل تاريخى يحتوى على تاريخ معين . أما الطرف الأيمن ("01/30/90") فهو بيان حرفى لوجود علامات التنصيص (Quotation Marks). ولعلاج ذلك يجب تحويل التاريخ (Date) الى القيمة الحرفية المناظرة كالآتى :

```
LIST FOR DTOC(Date) = "01/30/90"
```

٦- ٨ - ٢ الرسالة (Invalid function argument)

تظهر هذه الرسالة عادة عند استخدام نوع معين من البيانات مع دالة تستخدم نوعا آخر . فمثلا عند كتابة السطر التالى :

```
? UPPER(X)
```

وبفرض أن المتغير (X) يحتوى على قيمة عددية (Numeric) . فى هذه الحالة تظهر الرسالة :

```
Invalid function argument
```

وذلك لأن الدالة (UPPER) تستخدم فقط لتحويل المدخلات الحرفية الى حروف كبيرة (Uppercase) .

٦- ٨ - ٣ الرسالة (Unrecognized command verb)

تظهر هذه الرسالة عادة عند كتابة هجاء الأمر خطأ ، وفى هذه الحالة يتم مراجعة هجاء الأمر . كما يتم مراجعة هيئة هذا الأمر (Syntax). ويمكن استخدام شاشات المساعدة (Help) فى تحديد الهيئة (Syntax) الخاصة بكل أمر .

٦ - ٨ - ٤ الرسالة (Variable not found)

تظهر هذه الرسالة عادة عند كتابة اسم متغير ذاكرة لم يسبق تعريفه فى البرنامج . وفى هذه الحالة يمكن استخدام الأمر (DISPLAY MEMORY) لعرض محتويات الذاكرة ومعرفة أسماء المتغيرات الموجودة فى الذاكرة . كما يمكن استخدام الأمر (DISPLAY STRUCTURE) لعرض أسماء الحقول فى ملف قاعدة البيانات المفتوح .

٦- ٨ - ٥ الرسالة (Record out of range)

تظهر هذه الرسالة عند محاولة الذهاب الى سجل (Record) غير موجود فى الملف . فمثلا عند كتابة السطر التالى :

```
GOTO 99
```

بفرض ان الملف يحتوى على (98) سجلا فقط . فى هذه الحالة تظهر الرسالة المذكورة .

وتظهر هذه الرسالة أيضا عند حدوث خطأ فى ملف الفهرس المفتوح نتيجة عدم فتحه عند اجراء بعض التعديلات فى ملف قاعدة البيانات مثلا . وهذا يتم علاجه باعادة انشاء الفهرس باستخدام الأمر (REINDEX) .

٦- ٨ - ٦ الرسالة (Too Many Files Open)

تظهر هذه الرسالة عادة عند فتح عدد كبير من الملفات فى نفس

الوقت . مع عدم تغيير عدد الملفات فى ملف مواصفات النظام (Config.sys) ليسمح بفتح هذا العدد من الملفات . في هذه الحالة يتم تعديل ملف المواصفات وتخصيص عدد الملفات (Files) المطلوبة للبرامج . كما يمكن التحكم فى البرنامج لتقليل عدد الملفات المفتوحة فى كل وقت معين . ويتم ذلك عن طريق اغلاق كل ملف بمجرد انتهاء التعامل معه .

كما يمكن استخدام ملف الخطوات (Procedure file) فى الاستغناء عن بعض الملفات ، حيث يمكن أن يحتوى ملف الخطوات على عدد من البرامج بحد أقصى ٣٢ برنامجا بدلا من كتابة كل منها في ملف منفصل .


Converted by Tiff Combine - (no stamps are applied by registered version)



الجزء الثانى

نظام معلومات شئون الطلبة

* تصميم النظام
* البرنامج الرئيسى
* برنامج مسح السجلات



الفصل السابع

تصميم النظام



٧ - ١ مقدمــة

الهدف من هذا البرنامج هو تصميم نظام معلومات للطلبة يمكن السيطرة عليه من خلال القوائم (Menu Driven) . وهذا يعنى أن الشخص الذى يقوم بتشغيل البرنامج لايحتاج الى معرفة أى شيء عن خصائص برنامج (DBase III+) وربما لايحتاج الى قدر كبير من المعلومات عن الحاسب . ولكن يمكنه عن طريق مجموعة من الاختيـارات الواضحة تنفيذ أى عمليات مطلوبة للحصول على أى معلومات أو تعديـل البيـانات الموجودة أو مسحها أو ... الخ .

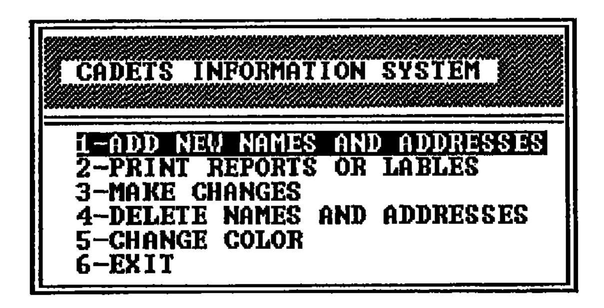
٧ - ٢ تصميم القائمة الرئيسية (Main Menu)

يتكون النظام فى هذه الحالة من أربعة برامج يتم ربطها والسيطرة عليها بواسطة برنامج خامس رئيسى . والبرنامج الرئيسى هو البرنامج الذى يقوم بعرض قائمة الاختيارات للمستخدم . وهذه القائمة تعتبر قلب النظام ، وهى أول ما يظهر أمام المستخدم وآخر ما يظهر أمامه ، كما يتم الرجوع اليها دائما بعد تنفيذ كل مهمة .

ويقوم المستخدم بتشغيل النظام بكتابة الأمر

(DO Cadets)

من مشيرة النقطة . حيث (Cadets) هو اسم البرنامج الرئيسى ، وفى هذه الحالة تظهر أمام المستخدم القائمة التالية :



شكل (٧ - ١)

وسوف يتم شرح كل اختيار من هذه الاختيارات والوظائف التى يؤديها قبل الدخول فى تفاصيل البرامج .

٧ - ٢ - ١ اضافة أسماء وعناوين جديدة

وهذا الاختيار يتيح للمستخدم اضافة سجلات جديدة . حيث يقوم المستخدم بكتابة الرقم (1) فتظهر أمامه الشاشة التالية .

```
CADET NO        ▮▮▮0   DATE ENTERED  ▮/▮/▮     TELEPHON ▮▮▮▮
NAME            ▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮  BLOOD TYPE ▮▮
CLASS           ▮▮▮   ADDRESS  ▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮
RELIGION        ▮▮▮▮▮▮▮  NATIONALITY ▮▮▮▮▮▮▮  BIRTH DATE ▮/▮/▮
BIRTH PLACE     ▮▮▮▮▮▮▮  FATHER NAME ▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮
MOTHER NAME     ▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮  FATHER SALARY ▮▮▮▮▮▮0
FATHER JOB      ▮▮▮▮▮▮▮▮  NO OF BROTHERS ▮▮▮0  SEC SCOOL AVG 0.00
DATE OF SEC. SCOOL ▮▮▮0  HOBBIES ▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮
NEAREST RELATIVE ,ADRESS ▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮
```

شكل (٧ - ٢)

وهى تشمل الحقول التى يتم ادخال البيانات فيها مع وجود مؤشر صغير على أول حقل . وبعد اضافة البيانات تعود القائمة الرئيسية للظهور مرة أخرى .

٧ - ٢ - ٢ طباعة التقارير والعناوين المختصرة

ويتم تنفيذ هذا الاختيار عندما يكتب المستخدم الرقم (٢) .وفى هذه الحالة تختفى القائمة الرئيسية وتظهر القائمة الموضحة بالشكل (٧ - ٣).

```
SELECT A REPORT CHOICE

1 - REPORT
2 - LABEls
3 - RETURN TO MAIN MENU
```

```
ENTER YOUR CHOICE<1-3> 3
```

شكل (٧ - ٢)

وعندما يختار المستخدم أحد هذه الاختيارات ، يظهر السؤال التالى :

Do you want (A)ll records, or (Q)uery?

واختيار (All) عن طريق كتابة الحرف (A) يؤدى الى عرض جميع بيانات الطلبة المخزنة في قاعدة البيانات . واختيار الاستفهام (Query) عن طريق كتابة الحرف (Q) يؤدى الى ظهور الرسالة التالية :

Enter name to display

مع ظهور عمود ضوئي (Highlight) يتم خلاله كتابة الحروف الأولى من الاسم المطلوب ، ويكفي في هذه الحالة كتابة أول حرف من الاسم فقط حيث يقوم البرنامج بعرض جميع الأسماء التى تبدأ بهذا الحرف مع أرقام السجلات الخاصة بها . كما يظهر أمام المستخدم السؤال التالى :

Which one do you want?

وهذا يتيح للمستخدم اختيار الاسم المطلوب عن طريق كتابة رقم السجل الخاص به .

وبعد اختيار المستخدم للاسم المطلوب عرض بياناته يظهر السؤال التالى :

```
Send report to the printer ? (Y/N)
```

فاذا كتب المستخدم الحرف (Y) يتم طباعة التقرير . وعند كتابة (N) يتم عرضه على الشاشة فقط .ويظهر فى الحالتيــن التقريــر الموضح بالشكل (٧ - ٤) .

CADET NO	:	6960
NAME	:	MOHAMED ALY SALEM
CLASS	:	55P
DATE OF ENTERING AIR ACADEMY	:	09/01/85
BLOOD CLASS	:	A
ADDRESS	:	13 - ABBAS ELAKKAD
TELEPHONE NO	:	2603556
RELIGION	:	MUSLIM
NATIONALITY	:	EGYPTIAN
HOBBIES	:	FOOTBALL
BIRTH DATE	:	01/22/67
BIRTH PLACE	:	TANTA
SECONDARY SCHOOL AVERAGE	:	70.00
DATE OF GETTING SECONDARY SCHOOL	:	1985
FATHER NAME	:	ALY SALEM
MOTHER NAME	:	FATMA MAHMOUD
FATHER SALARY	:	600
NOMBER OF BROTHERS	:	5

شكل (٧ - ٤)

٧ - ٢ - ٣ تعديل البيانات (Make Changes)

عندما يختار المستخدم الاختيار رقم (٢) من القائمة الرئيسية تختفى القائمة الرئيسية ويظهر الآتى على الشاشة :

```
Enter name of person to edit
or just press Return to Quit:
```

ويمكن للمستخدم ادخال أول حرف فقط من الاسم كما سبق الايضاح . وعندما يدخل المستخدم اسما غير موجود فى قاعدة البيانات يظهر الآتى على الشاشة :

```
There is no <name>
Press any key to try again
```

مع ملاحظة أن الاسم الخطأ الذى يدخله المستخـــــدم يظهـــر مكان (<name>) . وعندما يجد البرنامج هذا الاسم تظهر شاشة البيانات الخاصة به ويظهر المؤشر على أول حقل لتعديله حسب الحاجة .

وعندما يكون هناك أكثر من سجل بنفس الاسم أو مبتدئين بنفس الحرف الذى قام المستخدم بادخاله يقوم البرنامج بعرض الأسماء على المستخدم ومعها أرقام السجلات حتى يقوم باختيار رقم السجل المقابل للاسم المطلوب .

وبعد أن يقوم المستخدم باجراء التعديل المطلوب على السجل تعود القائمة الرئيسية للظهور من جديد .

٧ - ٢ - ٤ مسح السجلات

عندما يختار المستخدم الاختيار رقم (4) من القائمة الرئيسية ، فان البرنامج يسأل عن الاسم المراد مسح بياناته . ويقوم المستخدم بادخال الاسم المطلوب أو أول حرف منه فقط كما سبق الايضاح . وفى هذه الحالة تظهر أمام المستخدم الأسماء التى تشترك فى هذا الحرف ومعها أرقام السجلات حتى يقوم باختيار السجل المطلوب . وفى هذه الحالة يتيح البرنامج للمستخدم التأكد من رغبته فى مسح هذا السجل عن طريق عرض بيانات السجل أمام المستخدم وسؤاله اذا كان يريد مسح هذا السجل أم لا ، وذلك كالآتى :

```
Record#      name           address

1            Salem Emam     12-dwawin-Cairo

Delete this record?(Y/N)
```

وعندما يكتب المستخدم الحرف (Y) يظهر السؤال عن الاسم المراد مسح بياناته مرة أخرى لكى يدخل المستخدم اسما آخر اذا أراد . وعند انتهائه

من ادخال كل الأسماء التي يريد مسحها يمكنه الضغط على مفتاح الادخال بدلا من ادخال اسم جديد . وفي هذه الحالة يتيح له البرنامج التأكد مرة ثانية أنه يريد مسح جميع الأسماء التى أدخلها عن طريق عرض الشاشة التالية :

```
Records to be deleted..

Record#  name                 address
     6 * Eman Salem     12 - Dawawin - Cairo
     9 * Medhat Taher   10 - Nasr city - Cairo

Delete all these ? (Y/N)
```

وعندما يكتب المستخدم (N) . فان البرنامج يتيح له استعادة أي سجل من هذين السجلين حتى لايتم مسحه نهائيا . وتتكرر هذه العملية حتى يتأكد المستخدم تماما من السجلات التي يريد مسحها نهائيا . وفى هذه الحالة يقوم بكتابة (Y) أمام السؤال (Delete all these?) . فتختفى هذه السجلات تماما من قاعدة البيانات .

٧ - ٢ - ٥ الخروج من النظام (Exit)

عندما يختار المستخدم الاختيار رقم (5) فانه يخرج من البرنامج ويعود الى مشيرة النقطة (Dot Prompt) .

٧ - ٣ انشاء ملف قاعدة البيانات

يتم انشاء ملف قاعدة البيانات عن طريق قوائم برنامج المساعد (Assistant) أو عن طريق كتابة الأمر (CREATE Cadets) من مشيرة النقطة (Dot Prompt) . وذلك كما سبق الايضاح من خلال الكتاب الأول . حيث تظهر القائمة الموضحة بالشكل (٧ - ٥) والتي يتم من خلالها تحديد أسماء الحقول (Fields) ونوع كل حقل وعرضه وعدد الأرقام العشرية .

CURSOR <— —>	INSERT	DELETE	Up a field: ↑
Char: ← →	Char: Ins	Char: Del	Down a field: ↓
Word: Home End	Field: ^N	Word: ^Y	Exit/Save: ^End
Pan: ^← ^→	Help: F1	Field: ^U	Abort: Esc

	Field Name	Type	Width	Dec
1	NAME	Character	25	
2	JOB	Character	10	
3	ADDRESS	Character	30	
4	AGE	Numeric	2	0
5	NOTES	Memo	10	
6	PRICE	Numeric	7	2
7	QTY	Numeric	7	2

MODIFY STRUCTURE |<C:>| MOS1 | Field: 1/7

شكل (٧ - ٥)

ونفترض أن هيكل اللف كالآتى :

Field	Field Name	Type	Width	Dec
1	NAME	Character	35	
2	CLASS	Character	4	
3	ADDRESS	Character	40	
4	SEC_SCHOOL	Numeric	5	2
5	DT_ENT	Date	8	
6	T_NO	Character	8	
7	BLOOD	Character	3	
8	RELLIGION	Character	12	
9	NATION	Character	10	
10	B_DATE	Date	8	
11	B_PLACE	Character	12	
12	FATH_NAME	Character	32	
13	MOTH_NAME	Character	33	
14	F_SALARY	Numeric	10	

15	F_JOB	Character	14
16	NO_BROTHER	Numeric	5
17	DATE_SEC	Numeric	5
18	HOBBIES	Character	38
19	N_RELATIVE	Character	35
20	NOTES	Memo	10

ويمكن انشاء ملف الفهرس أيضا عن طريق برنامج المساعد (Assistant) كما سبق الايضاح . أو عن طريق كتابة الأمر التالى من مشيرة النقطة :

```
INDEX ON UPPER(name) TO Name
```

ويؤدى هذا الأمر الى انشاء الملف الفهرسى (Name.ndx)

وتم استخدام الدالة (UPPER) هنا لتحويل الأسماء فى الفهرس الى حروف كبيرة . ويؤدى هذا الى توحيد شكل الأسماء داخل الفهرس بصرف النظر عن شكلها فى ملف قاعدة البيانات . وهذا يؤدى الى سرعة الوصول الى الاسم المطلوب كما سيتم الايضاح فيما بعد .

وبعد انشاء ملف قاعدة البيانات وملف الفهرس الخاص به ، يتم فتح هذين الملفين أو اغلاقهما من خلال البرنامج كما سيتم الايضاح . واذا أراد المستخدم تعديل قاعدة البيانات دون استخدام البرنامج فيجب فى هذه الحالة التأكد من فتح ملف الفهرس مع ملف قاعدة البيانات حتى يتم تحديث الفهرس تبعا لأى تعديل فى قاعدة البيانات . ويتم ذلك عن طريق كتابة الأمر التالى :

```
USE Cadets INDEX Name
```

وهذا يؤدى الى فتح ملف قاعدة البيانات (Cadets.dbf) وملف الفهرس (Name.ndx) فى نفس الوقت .

٧ - ٤ انشاء شاشة الادخال

يحتاج البرنامج الى تصميم شاشة ادخال (Screen) حتى يستطيع المستخدم عن طريقها ادخال البيانات الى ملف قاعدة البيانات . ويمكن انشاء شاشة الادخال عن

طريق قوائم المساعد (Assistant) كما سبق الايضاح . كما يمكن انشاؤها أيضا بواسطة برنامج يتم التفرع اليه من البرنامج الرئيسى . وكلا الطريقتين سبق شرحهما في الكتاب الأول ، ولذلك سنفترض هنا انشاء شاشة الادخال الموضحة بالشكل (٧ - ٢) ، ونفترض أن الملف الخاص بهذه الشاشة اسمه (Cadets.fmt) .

٧ - ٥ انشاء التقرير

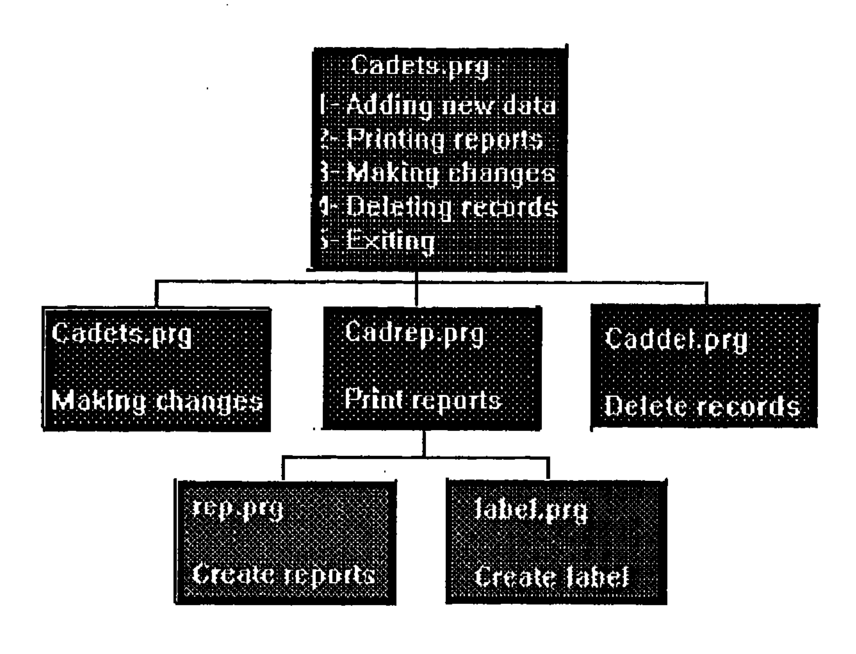
يمكن أنشاء التقرير عن طريق قوائم برنامج المساعد (Assistant) أيضا أو عن طريق الأمر (CREATE REPORT) . وفى الحالتين تظهر القوائم التى سبق شرحها والتى يتم عن طريقها تحديد شكل التقرير المطلوب طباعته . مع ملاحظة أن انشاء التقرير لايعنى كتابة أى بيانات من ملف قاعدة البيانات فيه ، ولكنه يحدد فقط الحقول المطلوب اظهارها فى التقرير ومواقع هذه الحقول وأطوالها ثم تظهر البيانات بعد ذلك عند استخدام هذا التقرير فى عرض بيانات سجلات معينة فى الملف .

وهناك طريقة أخرى لتصميم التقرير ، وذلك عن طريق برنامج فرعى يتم كتابته . وسوف نستخدم هذه الطريقة في انشاء التقرير لأنها تحقق مرونة كبيرة في توزيع الحقول على الشاشة والتحكم فى شكل التقرير . وهذا سيتم شرحه عند دراسة البرامج الفرعية المختلفة .

٧ - ٦ تركيب البرنامج

عند تصميم نظام يحتوى على عدة برامج فمن المهم فى هذه الحالة رسم الشكل الهرمى (Hierarchical Structure) لهذه البرامج . ومع أن ذلك قد يبدو غير مطلوب مع هذا النظام ، لأنه لايحتوى على العديد من البرامج ، فانه من الأفضل دائما التعود على رسم الشكل الهرمى حتى يمكن تتبع كل برنامج على حدة ومعرفة البرنامج الذى يقوم بالاستدعاء (Calling Program) والبرنامج الذى يتم استدعاؤه (Called Program) .

أنظر الشكل (٧ - ٦) .



شكل (٧ - ٦)

ويتضح من هذا الشكل الهرمى أن النظام قد تم تقسيمه الى عدة وظائف رئيسية، وكل وظيفة من هذه الوظائف ينفذها برنامج معين . كما يتضح أيضا أن البرنامج (Cadets.prg) هو البرنامج الذى يقع على قمة الهرم ، وبالتالى فانه يسيطر على باقى البرامج .

كما أن كل برنامج فرعى بعد انتهاء مهمته يعود الى البرنامج الرئيسى ليقوم بتوزيع باقى المهام .

وعملية تقسيم النظام الى وظائف صغيرة يتم تنفيذها من خلال برامج فرعية تسهل تصميم النظام الى درجة كبيرة جدا . حيث يصبح من السهل تصميم وتطوير واختبار كل برنامج صغير على حدة . كما أن ذلك يوفر على مخطط البرامج البحث فى عدد كبير من سطور البرنامج عندما يريد تعديل أى جزء من البرنامج أو تصحيحه . حيث يمكنه فى هذه الحالة تحديد المهمة أو الوظيفة التى يريد تعديلها ثم يقوم بتعديل البرنامج المسئول عن تنفيذ هذه المهمة .

الفصل الثامن

البرنامج الرئيسى



يفضل دائما قبل البدء فى تصميم البرنامج كتابة الخطوات المطلوبة باللغة الواضحة بالنسبة لمخطط البرامج سواء كانت اللغة الانجليزية أو العربية أو أى لغة أخرى ، وتسمى هذه الخطوات (PSEUDOCODE) . وفى البرنامج الرئيسى الخاص بالطلبة (Cadets) تكون هذه الخطوات كالآتى مثلا :

١- تجهيز بيئة البرنامج (Environment) .
٢- فتح ملف قاعدة البيانات وملف الفهرس المصاحب له .
٣- انشاء حلقة تكرارية لعرض القائمة الرئيسية .
٤- مسح الشاشة .
٥- عرض القائمة الرئيسية كالآتى :

```
CADETS INFORMATION SYSTEM

1-ADD NEW NAMES AND ADDRESSES
2-PRINT REPORTS OR LABLES
3-MAKE CHANGES
4-DELETE NAMES AND ADDRESSES
5-CHANGE COLOR
6-EXIT

HILIGHT OPTION BY USING ↑ OR ↓ AND PRESS ↵
OR PRESS APPROPRIATE MENU NUMBER
```

شكل (٨ - ١)

٦- استقبال اختيار المستخدم .
٧- التفرع الى الأمر أو البرنامج الذى يحقق اختيار المستخدم .
٨- اعادة عرض القائمة الرئيسية فى حالة عدم اختيار الخروج (Exit) .
٩- الرجوع الى مشيرة النقطة فى حالة اختيار الخروج (Exit) .

وبعد الانتهاء من كتابة الخطوات الأولية (PSEUDOCODE) يتم كتابة البرنامج الذى يحقق هذه الخطوات كالآتى :

```
****************** Cadets.prg
****************** Cadets Information System

* - - - - - - - - - - - Set up initial parameters
SET TALK OFF
SET STATUS OFF
SET DEFAULT TO C

* - - - - - - - - - Open the database and index files
USE CADETS INDEX Name

* - - - - - - - - - - - Begin loop for main menu
choice = 0
DO WHILE choice <> 5
     CLEAR

       TEXT

             Cadets Information System

          1 -  Add new names and addresses
          2 -  Print Reports
          3 -  Make changes
          4 -  Delete names and addresses
          5 -  Exit

       ENDTEXT

* - - - - - - - - - - - Get users choice
@ 16,20 SAY "Enter choice: " GET choice Picture "9";
RANGE 1,5
READ

    * - - - - - - Branch to appropriate task orprogram.
    DO CASE
        CASE choice = 1
        SET FORMAT TO Cadets
        APPEND
```

```
        CLOSE FORMAT
    CASE choice = 2
        DO cadrep
    CASE choice = 3
        DO Cadedit
    CASE choice = 4
        DO Caddel
    ENDCASE
ENDDO (while choice <>5)

* - - - - - - - - - - - - -  Returning to the dot prompt
SET TALK ON
SET STATUS ON
RETURN
```

ويمكن شرح اجزاء البرنامج كالآتى :

يبدأ البرنامج بعرض اسم البرنامج ووظيفته ثم أوامر تجهيز بيئة البرنامج مثل الأمر (SET TALK ON) الذى يمنع ظهور خطوات التشغيل على الشاشة ، ثم الأمر (SET STATUS OFF) الذى يمسح عمود الحالة (Status Bar) من أسفل الشاشة أثناء تنفيذ البرنامج ، ثم الأمر (SET DEFAULT TO C) الذى يجعل القرص الصلب هو القرص المستخدم (Default Drive) . ويمكن تغيير هذا الأمر اذا كان البرنامج موجودا على قرص مرن باستبدال الحرف (C) بالحرف (A) أو الحرف (B) حسب الحاجة .

والجزء الثانى من البرنامج يفتح ملف قاعدة البيانات (Cadets.dbf) وملف الفهرس المصاحب له (Name.ndx) .

والجزء الثالث يقوم بانشاء الحلقة التكرارية التى يستمر تنفيذها حتى يكتب المستخدم الرقم (5) للخروج من البرنامج ، ويتم من خلال هذه الحلقة التكرارية عرض القائمة الرئيسية للبرنامج . والأمر (TEXT) والأمر (ENDTEXT) يعرضان جميع السطور المحصورة بينهما والخاصة بقائمة الاختيارات كما سبق الايضاح .

ويلاحظ فى بداية الحلقة التكرارية انشاء متغير الذاكرة (choice) واعطاؤه القيمة صفر .

والجزء الرابع يقوم باستقبال اختيار المستخدم وتخزينه فى المتغير (choice) الذى سبق انشاؤه . ويلاحظ استخدام الاوامر (@, SAY, GET, READ) . كما يلاحظ استخدام عبارة (Picture"9") . واستخدام الرقم (9) هنا يجبر المستخدم على ادخال أعداد وليس حروف . كما يلاحظ استخدام عبارة (RANGE 1,5) . وهذا يجبر المستخدم على ادخال رقم بين (5,1) لأن أى رقم آخر لن يقبله البرنامج .

والجزء الخامس يستخدم الأمر (DO CASE) الذى يساعد البرنامج على اتخاذ القرار بناء على اختيار المستخدم . وعندما يختار المستخدم الرقم (1) فان البرنامج يفتح شاشة الادخال الخاصة بملف قاعدة البيانات المفتوح . وذلك من خلال الأمر :

```
SET FORMAT TO Cadets
```

كما يسمح للمستخدم باضافة سجل جديد عن طريق الأمر (APPEND) . وبعد الانتهاء من ادخال هذا السجل يتم اغلاق ملف شاشة الادخال عن طريق الأمر (CLOSE FORMAT) . وفى هذه الحالة تظهر القائمة الرئيسية مرة أخرى .

والاختيارات الأخرى (2, 3, 4) تؤدى الى التفرع الى برامج فرعية خارجية .

والجزء السادس والأخير من البرنامج يشكل أوامر نهاية الحلقة التكرارية التى يتم تنفيذها فى حالة اختيار المستخدم للاختيار (5) . ويلاحظ اضافة تعليق (Comment) بجوار الأمر (ENDDO) يوضح وظيفة الحلقة التكرارية وذلك لأن أى تعليق (Comment) يتم اضافته بعد الأمر (ENDDO) أو الأمر (ENDIF) لايؤثر فى تنفيذ البرنامج . وهذا يكون مفيدا بصفة خاصة عند وجود عدة حلقات تكرارية متداخلة ، حيث يوضح التعليق (Comment) أى (ENDDO) تتبع أى (DO WHILE) .

ثم يلاحظ بعد انتهاء الحلقة التكرارية استخدام الأوامر التى تعيد بيئة الحاسب الى ما كانت عليه قبل البرنامج . وهى الأوامر التالية :

```
SET TALK ON
SET STATUS ON
RETURN
```

ملاحظة

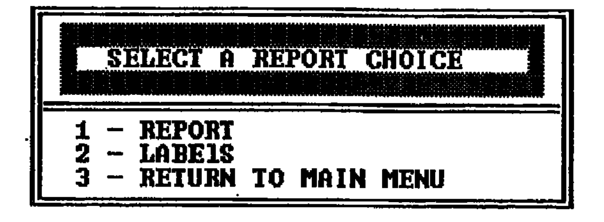
تجدر الاشارة الى أن المستطيلات التى تظهر فى شاشة القائمة الرئيسية و كذلك العمود المتحرك تتطلب اضافة برنامج فرعى خاص (Routine) . و لتبسيط شرح البرنامج للمبتدئين فقد تم حذف هذا البرنامج الفرعى . كما تم شرح هذا البرنامج الفرعى فى نهاية الكتاب للرجوع اليه عنـــدما يراد اضافة أى مؤثرات خاصة فى أى قائمة .

الفصل التاسـع

برنامج التقارير

كما سبق الايضاح فان برنامج التقارير هو البرنامج (Cadrep.prg) الذى يتم التفرع اليه من البرنامج الرئيسى فى حالة اختيار المستخدم للاختيار رقم (2) . وهذا البرنامج يقوم بعرض قائمة اختيارات فرعية تشمل ثلاثة اختيارات لتحديد نوع التقرير المطلوب أو الرجوع الى القائمة الرئيسية . كما يظهر سؤال عما اذا كان المطلوب عرض التقرير على الشاشة أو طباعته على الطابعة . ويتم توضيح الخطوات الأولية (PSEUDOCODE) كالآتى :

١ - مسح الشاشة وعرض شاشة نوع التقرير المطلوب كالآتى :



ENTER YOUR CHOICE(1-3) 3

شكل (٩ - ١)

٢ - فى حالة عدم اختيار أى نوع يتم الرجوع الى القائمة الرئيسية مرة ثانية .
٣ - السؤال عما اذا كان المطلوب عرض جميع بيانات قاعدة البيانات فى التقرير أو البحث (Query) عن سجل معين وعرضه .
٤ - عند اختيار البحث (Query) يتم سؤال المستخدم عن الاسم المطلوب البحث عنه .
٥ - يتم البحث عن السجل المطلوب .
٦ - يتم حصر عدد السجلات التى تحتوى على نفس الاسم .
٧ - فى حالة عدم العثور على أى سجل يحتوى على الاسم المطلوب يتم تنبيه المستخدم حتى يدخل اسما آخر .
٨ - فى حالة وجود عدة سجلات تحتوى على نفس الاسم يتم عرض هذه السجلات على المستخدم لاختيار أحدها عن طريق رقم السجل .
٩ - يتم السؤال عما اذا كان المطلوب عرض التقرير على الشاشة أم طباعته على الطابعة .
١٠- يتم طباعة التقرير .
١١- عند اختيار تقرير (Report) يتم التفرع الى برنامج (Rep.prg) .
١٢- عند اختيار تقرير مختصر (Label) يتم التفرع الى برنامج (Label.prg) .

١٣- ايقاف الشاشة مؤقتا (Pause) فى حالة عرض التقرير على الشاشة حتى يستطيع المستخدم قراءة هذا التقرير .
١٤- العودة الى القائمة الرئيسية مرة ثانية .

بعد الانتهاء من كتابة الخطوات الأولية (PSEUDOCODE) يتم كتابة البرنامج كالآتى :

```
* * * * * * * * * * *Cadrep.prg
* - -  Reports program for Cadets Information system.

* - - - - - - - - Clear screen and ask about report type
CLEAR
TEXT
                         Select a report type

                   1 -   Report
                   2 -   Label
                   3 -   Return to main menu

ENDTEXT

* - - - Initialize variable and ask form report type.
repchoice = 0
15,20 SAY "Enter your choice (1-3)" GET repchoice;
        PICTURE "9" RANGE 1,3
READ

* - - - - - - - - If return chosen, return to main menu.
IF repchoice = 3
        RETURN
ENDIF

* - - - - - - - - Ask about query.
CLEAR
qchoice = "A"
@ 10,16 SAY "Do you want (A)ll records,or a (Q)uery?";
        GET qchoice PICTURE "!"
READ
```

```
* - - - - - - - making Query if requested
SET EXACT OFF
IF qchoice = "Q"
SET EXACT OFF
qname = SPACE(8)
@ 16,16 SAY "Enter name to display" GET qname
READ
qname = UPPER(TRIM(qname))
SEEK qname
recnumb = RECNO()
COUNT WHILE UPPER(name) = qname TO howmany


* - - - - -  IF the name is not found, warn the user
IF howmany = 0
      @ 19,15 SAY "There is no & qname"
      ? CHR (7)
      WAIT "Press any key to return to main menu"
      RETURN
ENDIF(howmany = 0)

* - - - - IF there is more than one record having the
* - - - - same name , they are displayed to the user .
IF howmany > 1
    CLEAR
    GO TOP
    SEEK qname
    DISPLAY name, class WHILE UPPER(name) = qname
    @ ROW()+2,15 SAY "which one do you want";
    GET recnumb
    READ
    GOTO recnumb
ELSE
    SET FILTER TO UPPER(name) = qname
ENDIF(howmany>1)
ENDIF(qchoice = Q)
```

```
* - - - - - - - Ask about printer
toprint = "N"
CLEAR
@ 10,5 SAY "Send records to printer? Y/N" GET toprint;
PICTURE "!"
READ

* - - - - - Displaying the report
IF toprint = "Y"
    SET DEVICE TO PRINT
ELSE
    SET DEVICE TO SCREEN
ENDIF
DO CASE
    CASE repchoice = 1
        DO rep
    CASE repchoice = 2
        DO label
ENDCASE

        * - - - - - If printer was not selected ,pause
        * - - - - - before returning to menu
        IF toprint <> "Y"
            @ 24,1 CLEAR
            WAIT "press any key to return to menu ..."
        ENDIF
* - - - - - - When report is done, close filter,
* - - - - - - and return to main menu
SET FILTER TO
RETURN
```

والبرنامج يبدأ كالعادة بكتابة اسم البرنامج . ثم يبدأ الجزء الأول بمسح الشاشة وعرض قائمة اختيارات نوع التقرير المطلوب . ثم يقوم الجزء الثاني بانشاء متغير الذاكرة (repchoice) وتخزين القيمة صفر فيه . وهذا يجعل المتغير عدديا (numeric) . ثم يتم عرض رسالة للمستخدم (prompt) لادخال الاختيار المطلوب وتخزينه في التغير (repchoice) . ويتم استخدام الصورة (PICTURE"9" لاجبار المستخدم على ادخال أعداد . كما يستخدم المدى (RANGE 1,3 لتحديد مدى الارقام الذى يمكن ادخاله بحيث لايزيد عن

(٣) وهذا الجزء يحتوى على السطور التالية :

```
* - - - - - Initialize variable and ask for report type
repchoice = 0
15,20 SAY "Enter your choice (1-3)" ;
GET rephoice PICTURE "9" RANGE 1,3
READ
```

ويجب ملاحظة أنه عندما يزيد طول الأمر عن عرض الشاشة يتم كتابة الحرف (;) فى نهاية السطر واستكمال كتابة الأمر فى السطر التالى .

والجزء الثالث من البرنامج يؤدى الى العودة الى القائمة الرئيسية فى حالة كتابة المستخدم الرقم (3) . وهو يحتوى على السطور التالية :

```
IF repchoice = 3
    RETURN
ENDIF
```

والجزء الرابع يسأل المستخدم اذا كان يريد عرض جميع سجلات الملف فى التقرير أو يريد البحث عن سجل معين وعرض بياناته . ويتم تخزين اختيار المستخدم فى المتغير (qchoice) . وهذا الجزء يتكون من السطور التالية :

```
qchoice = "A"
@ 10,16 SAY "Do you want (A)ll records,or a(Q)uery?" ;
GET qchoice PICTURE "!"
READ
```

والجزء الخامس يقوم بالبحث عن الاسم الذى يدخله المستخدم فى حالة اختيار الحرف (Q) . وفى هذه الحالة يتم عرض رسالة للمستخدم لادخال الاسم المطلوب البحث عنه ثم تخزين هذا الاسم فى المتغير (qname) . ويتم تحويل هذا الاسم الى حروف كبيرة عن طريق الدالة (UPPER) . كما يتم مسح المسافات الزائدة عن طريق الدالة (TRIM) . ويمكن للمستخدم ادخال حرف واحد أو أكثر من الاسم .

ويتم البحث بواسطة الأمر (SEEK) ثم تخزين رقم السجل الذى يتم الوصول اليه فى المتغير (recnumb) ثم يتم حصر عدد السجلات التى تشترك فى هذا الاسم أو الحرف الذى يتم ادخاله ، وتخزين هذا العدد فى المتغير (howmany) .

ويستخدم الأمر (SET EXACT OFF) حتى تكون المقارنة بين الاسم أو الحرف الذى يتم ادخاله وبين حقل الاسم (name) غير كاملة . أى يكفى وجود الحروف فى أول الاسم لتحقيق شرط البحث . ومع أن الوضع المبدئى (Default) هو (SET EXACT OFF) الا أنه يفضل كتابته فى البرنامج خشية أن يكون قد تم تغيير وضعه المبدئى قبل تنفيذ البرنامج .

ويتكون هذا الجزء من السطور التالية :

```
IF qchoice = "Q"
    SET EXACT OFF
    qname = SPACE(8)
    @ 16,16 SAY "Enter name to display" GET qname
    READ
    qname = UPPER(TRIM(qname))
    SEEK qname
    recnumb = RECNO()
    COUNT WHILE UPPER(name) = qname TO howmany
```

والجزء السادس من البرنامج يقوم بتحذير المستخدم فى حالة عدم العثور على الاسم المطلوب . ويلاحظ هنا استخدام دالة الماكرو فى عرض الاسم أو الحرف الذى أدخله المستخدم داخل السلسلة الحرفية . كما يلاحظ استخدام الداله CHR(7) لتشغيل جرس التنبيه . ثم تظهر رسالة للمستخدم ليضغط على أى مفتاح للرجوع الى القائمة الرئيسية مرة أخرى . ويحتوى هذا الجزء على السطور التالية :

```
IF howmany = 0
    @ 19,15 SAY "There is no & qname"
    ? CHR (7)
    WAIT "Press any key to return to main menu
    RETURN
ENDIF(howmany = 0)
```

والجزء السابع يقوم بعرض بيانات الاسم والفصل الدراسى الخاص بالاسماء التى تشترك فى الاسم أو الحرف المطلوب حتى يقوم المستخدم باختيار رقم السجل الخاص بالاسم المطلوب من هذه السجلات . ويلاحظ أن أمر البحث قد تم استخدامه مرة ثانية ، وذلك لأن مؤشر السجلات (Record Pointer) يكون قد تحرك نتيجة استخدام الأمر (COUNT) ويراد اعادته مرة ثانية الى أول سجل يحقق الشرط . ويتم أولا تحريك المؤشر الى أول الملف عن طريق الأمر (GO TOP) حتى يبدأ البحث من

أول الملف . ثم يتم عرض بيانات السجلات عن طريق الأمر (DISPLAY) . ويظهر سؤال للمستخدم عن رقم السجل الذى يراد عرض بياناته فى التقرير ويتم تخزين هذا الرقم فى المتغير (recnumb) . ثم يتم الذهاب الى هذا السجل استعدادا لطباعته .

وفى حالة العثور على سجل واحد يحقق الشرط أى أن المتغير (howmany) تكون قيمته (١) . أى فى حالة عدم تحقق الشرط (howmany>1) . فى هذه الحالة يتم استخدام مرشح (Filter) للحصول على السجل الذى يحقق الشرط .

وهذا الجزء يحتوى على السطور التالية :

```
IF howmany > 1
    CLEAR
    GO TOP
    SEEK qname
    DISPLAY name, class WHILE UPPER(name) = qname
    @ ROW()+2,15 SAY "which one do you want";
    GET recnumb
    READ
    GOTO recnumb
ELSE
    SET FILTER TO UPPER(name) = qname
ENDIF(howmany>1)
ENDIF(qchoice = Q)
```

والجزء الثامن يسأل المستخدم اذا كان يريد طباعة التقرير أو الاكتفاء بعرضه على الشاشة . ويتم تخزين اجابة المستخدم فى المتغير (toprint) . ويحتوى هذا الجزء على السطور التالية :

```
toprint = "N"
CLEAR
@ 10,5 SAY "Send records to printer? Y/N" GET toprint;
PICTURE "!"
READ
```

والجزء التاسع يؤدى الى توجيه التقرير الى الشاشة أو الطابعة حسب اختيار المستخدم الموجود فى المتغير (toprint) . فاذا كان هذا الاختيار (Y) يتم

توجيهه الى الطابعة ، واذا كان غير ذلك يتم توجيهه الى الشاشة . ثم يتم تنفيذ البرنامج (rep.prg) عندما يكون المتغير (repchoice) محتويا على الرقم (1) . كما يتم تنفيذ البرنامج (label.prg) عندما يكون المتغير (repchoice) محتويا على الرقم (2) . وهذا الجزء يحتوى على السطور التالية :

```
IF toprint = "Y"
    SET DEVICE TO PRINT
ELSE
    SET DEVICE TO SCREEN
ENDIF
DOCASE
    CASE repchoice = 1
        DO rep
    CASE repchoice = 2
        DO label
ENDCASE
```

والجزء العاشر يتم من خلاله ايقاف التقرير على الشاشة مؤقتا (Pause) حتى يستطيع المستخدم قراءة بيانات التقرير ثم يضغط على أى مفتاح للرجوع الى قائمة التقارير مرة ثانية . ويحتوى هذا الجزء على السطور التالية :

```
IF toprint <> "Y"
    @ 24,1 CLEAR
    WAIT "Press any key to return to menu"
ENDIF
```

والجزء الحادى عشر يتم من خلاله اغلاق المرشح (Filter) حتى يعود الملف الى حالته الأولى . ثم يتم الرجوع الى القائمة الرئيسية باستخدام الأمر (RETURN) .

٩ - ١ البرنامج (rep)

يتم كتابة هذا البرنامج بديلا عن استخدام قوائم برنامج المساعد (Assistant في تصميم شكل التقرير . حيث يتم تصميم التقرير باستخدام جموعة من الأوامر (@...SAY) داخل حلقة تكرارية يتم تكرارها حتى نهاية سجلات لف (EOF) . وذلك لعرض التقارير كلها فى حالة طلب المستخدم ذلك . وهذا البرنامج يكون كالآتى :

```
DO WHILE .NOT. EOF()
CLEAR
   @  2, 2 SAY "Cadet name :"
   @  2,37 SAY NAME
   @  3, 2 SAY "CLASS :"
   @  3,38 SAY CLASS
   @  4, 2 SAY "ADDRESS :"
   @  4,38 SAY ADDRESS
   @  5, 2 SAY "Secondary school average :"
   @  5,38 SAY SEC_SCHOOL
   @  6, 2 SAY "Date of entering the institute :"
   @  6,38 SAY DT_ENT
   @  7, 2 SAY "Telephone no :"
   @  7,38 SAY T_NO
   @  8, 2 SAY "Blood Type:"
   @  8,32 SAY BLOOD
   @  9, 2 SAY "Religion   :"
   @  9,38 SAY RELIGION
   @ 10, 2 SAY "Nationality :"
   @ 10,38 SAY NATION
   @ 11, 2 SAY "Birth Date :"
   @ 11,38 SAY B_DATE
   @ 12, 2 SAY "Birth place :"
   @ 12,38 SAY B_PLACE
   @ 13, 2 SAY "Father name:"
   @ 13,38 SAY FATH_NAME
   @ 14, 2 SAY "Mother name:"
   @ 14,38 SAY MOTH-NAME
   @ 15, 2 SAY "Father salary:"
   @ 15,38 SAY F_SALARY
   @ 16, 2 SAY "Father Job:"
   @ 16,38 SAY F_JOB
   @ 17, 2 SAY "Number of brothers :"
   @ 17,38 SAY NO_BROTHER
   @ 18, 2 SAY "Date of getting Sec-School:"
   @ 18,38 SAY DATE_SEC
   @ 19, 2 SAY "Hobbies"
   @ 19,38 SAY HOBBIES
```

```
@ 20, 2 SAY "Nearest relatives"
@ 20,38 SAY N_RELATIVE

WAIT "Press any key to continue"
CLEAR "
@ 1,40 SAY "Notes"
DISPLAY OFF NOTES

* - - - Ask the user if he want to stop displaying
KEYPRESS = 0
@ 22,12 SAY "Press any key to continue or (Q) to stop
DO WHILE KEYPRESS = 0
     KEYPRESS = INKEY()
ENDDO(KEYPRESS)
IF KEYPRESS = 113 .OR. KEYPRESS = 81
     CLEAR
     RETURN
ENDIF
SKIP
ENDDO(while not EOF)
```

ويلاحظ أن هذا البرنامج يحتوى على سطرين لكل حقل . السطر الأول يتم بواسطته عرض اسم هذا الحقل . لذلك يتم كتابة الاسم كسلسلة حرفية (string) . والسطر الثاني يتم بواسطته عرض محتويات هذا الحقل . كما يلاحظ من الاحداثيات (X,Y) التي تلي الحرف (@) أن اسم الحقل ومحتويات الحقل يتم عرضهما على سطر واحد على الشاشة . لذلك فعند عرض سجل معين على الشاشة تظهر الشاشة الموضحة بالشكل (٩ - ٢) مثلا .

كما يلاحظ أيضا عرض حقل الملاحظات (Notes) فى شاشة مستقلة . وذلك بمسح الشاشة السابقة عند ضغط المستخدم على أى مفتاح .

والجزء الثاني من البرنامج الذي يبدأ بانشاء المتغير (KEYPRESS) يقوم بعرض رسالة للمستخدم لايقاف عرض السجلات فى حالة عرض السجلات كلها. وذلك عندما يريد أن يوقف عرض هذه السجلات ويعود الى قائمة التقارير مرة ثانية .

ويلاحظ استخدام الدالة (INKEY()) فى اختبار الشفرة الخاصة بالمفتاح الذى ضغط عليه المستخدم . فاذا كانت هذه الشفرة تساوى (81) أو (113) فان هذا

يعنى أن المستخدم قد ضغط على مفتاح (Q) أو مفتاح (q) على الترتيب . وفى هذه الحالة يتم الخروج من البرنامج والعودة الى قائمة التقارير . أما اذا ضغط على أى مفتاح آخر ، فان هذا يؤدى الى عرض بيانات السجل التالى .

CADET NO	:	6960
NAME	:	MOHAMED ALY SALEM
CLASS	:	55P
DATE OF ENTERING AIR ACADEMY	:	09/01/85
BLOOD CLASS	:	A
ADDRESS	:	13 - ABBAS ELAKKAD
TELEPHONE NO	:	2603556
RELIGION	:	MUSLIM
NATIONALITY	:	EGYPTIAN
HOBBIES	:	FOOTBALL
BIRTH DATE	:	01/22/67
BIRTH PLACE	:	TANTA
SECONDARY SCHOOL AVERAGE	:	70.00
DATE OF GETTING SECONDARY SCHOOL	:	1985
FATHER NAME	:	ALY SALEM
MOTHER NAME	:	FATMA MAHMOUD
FATHER SALARY	:	600
NOMBER OF BROTHERS	:	5

شكل (٩ - ٢)

٩ - ٢ البرنامج (label)

يتم كتابة هذا البرنامج لتصميم التقارير المختصرة بدلا من استخدام برنامج المساعد (Assistant) . وهو يماثل البرنامج (rep) السابق شرحه ولذلك سوف نكتفى بكتابة سطور البرنامج دون شرحها وهى كالآتى :

```
* - - - - - label.prg
* - - - - Program for making labels
R = 2
GO TOP
DO WHILE .NOT. EOF()
```

```
    @ R,2   SAY CADET_NO
    @ R+1,2 SAY NAME
    @ R+2,2 SAY GRL_SECT
    @ R+3,2 SAY ADDRESS
    @ 17,2 SAY "Press any key to continue or + ;
    "(Q) to stop"
    KEYPRESS = 0
    DO WHILE KEYPRESS = 0
        KEYPRESS = 0
    ENDDO(KEYPRESS)
    IF KEYPRESS = 113 .OK. KEYPRESS = 81
        RETURN
    ENDIF
    CLEAR
    SKIP
    ?
ENDDO
RETURN
```

الفصل العاشر

برنامج التصحيح

يساعد هذا البرنامج على وصول المستخدم الى السجل المطلوب وتصحيحه .
وكالعادة يتم أولا كتابة الخطوات الأولية (PSEUDOCODE) كالآتى :

١ - يتم تكوين حلقة تكرارية لتعديل السجلات .
٢ - يتم السؤال عن الاسم أو الحرف المطلوب البحث عنه .
٣ - فى حالة عدم ادخال أى اسم ، يتم الرجوع الى القائمة الرئيسية .
٤ - يتم تحويل الاسم الى حروف كبيرة حتى يماثل ملف الفهرس .
٥ - يتم البحث عن الاسم المطلوب .
٦ - يتم حساب عدد السجلات التى تحتوى على نفس الاسم .
٧ - فى حالة عدم العثور على أى سجل يحتوى على هذا الاسم يتم تحذير المستخدم حتى يدخل اسما آخر .
٨ - فى حالة وجود عدة سجلات تحتوى على نفس الاسم يتم عرض هذه السجلات على المستخدم لاختيار أحدها عن طريق رقم السجل .
٩ - عند الوصول الى السجل المطلوب يتم عرض بياناته على شاشة الادخال التى سبق تصميمها .
١٠- يتم العودة الى القائمة الرئيسية بعد تعديل الحقول المطلوبة .

ثم يتم كتابة البرنامج كالآتى :

```
* * * * * * *  Cadedit.prg
* Lookup and edit data in the cadets database

* - - - - - - - Set up a loop for editing records
entering = .T.
DO WHILE entering
    SET EXACT OFF
    GO TOP
    * - - - - - - - Ask for name of person to lookup
    CLEAR
    lookup = SPACE(4)
    @ 10,12 SAY " Enter name of person to edit"
    @ 12,12 SAY "or just press Return to exit" ;
    GET lookup
    READ

    * - - - - - -  If no name entered , skip all
    * - - - - - -  commands between here and enddo
    IF lookup = "  "
        entering = .F.
```

```
        LOOP
    ENDIF

    * - - - - - - convert lookup to uppercase to
    * - - - - - - match index file, and trim it
    lookup = UPPER (TRIM(lookup))

SEEK lookup
    mrecord = RECNO()
    * - - - - - -  count how many there are
    COUNT WHILE UPPER (NAME) = lookup TO howmany
    IF howmany = 0
        @ 20,10 SAY "There is no & lookup"
        @ 22,10 SAY "press any key to try again"
        ? CHR(7)
        WAIT
        mrecord = 0
    ENDIF(howmany = 0)
    * - - - - - -  if more than one record has that
    * - - - - - -  name, get more information
    IF howmany > 1
        CLEAR
        mrecord = 0
        SEEK lookup
        DISPLAY NAME, CLASS WHILE UPPER(NAME) = lookup
        @ ROW()+3,10 SAY "Edit which record # ?" GET ;
        mrecord PICTURE "9999"
        READ
    ENDIF
    * - - - - - - -  If there is a record number greater
    * - - - - - - -  than zero, edit the record
    IF mrecord > 0
        CLEAR
        GOTO mrecord
        SET FORMAT TO cadets
        READ
        CLOSE FORMAT
    ENDIF
```

```
ENDDO (WHILE entering)
RETURN
```

ويبدأ هذا البرنامج باسم البرنامج ووظيفته كالمعتاد . ثم يتم تكوين حلقة تكرارية تعتمد على التغير المنطقي (entering) . وحيث أن قيمته فى البداية تكون (.T.) أى صحيح ، لذلك يتم تنفيذ الحلقة التكرارية أول مرة . ثم يتم التحكم من داخل الحلقة فى قيمة المتغير المنطقى (entering) لاستمرار تنفيذ الحلقة أو ايقاف تنفيذها .

وهذا الجزء يتكون من السطور التالية :

```
entering = .T.
DO WHILE entering
```

وفى الجزء الثانى من البرنامج يبدأ تنفيذ الحلقة التكرارية . ويتم انشاء المتغير (lookup) الذى يكون طوله (4) حروف . وذلك حتى يدخل المستخدم فيه الحروف الأولى من الاسم المطلوب البحث عنه . ثم يتم عرض رسالة للمستخدم لادخال الحروف المطلوبة وتخزين هذه الحروف فى التغير (lookup) . ويحتوى هذا الجزء على السطور التالية :

```
CLEAR
lookup = SPACE(4)
@ 10,12 SAY " Enter name of person to edit"
@ 12,12 SAY "or just press Return to exit" ;
GET lookup
READ
```

وفى الجزء الثالث يتم تخزين القيمة (.F.) فى المتغير (entering) فى حالة ضغط المستخدم على مفتاح الادخال دون كتابة أى حروف . وفى هذه الحالة يتم الانتقال الى أول الحلقة التكرارية بواسطة الأمر (LOOP) . وحيث أن المتغير (entering) يكون غير صحيح (False) ، فلا يتم تنفيذ الحلقة التكرارية وتعود القائمة الرئيسية للظهور . ويحتوى هذا الجزء على السطور التالية :

```
    IF lookup = "    "
    entering = .F.
    LOOP
    ENDIF
```

وفي الجزء الرابع من البرنامج يتم البحث عن الحروف التى يدخلها المستخدم خلال حقل الاسم (name) . مع ملاحظة أنه سبق فتح ملف الفهرس من خلال البرنامج الرئيسى . كما أن هذا الفهرس قد تم انشاؤه بحيث يتضمن الأسماء بحروف كبيرة (Upercase) حتى يسهل مطابقته على الاسم الذى يدخله المستخدم . كما يتم تحويل الحروف التى يدخلها المستخدم أيضا الى حروف كبيرة . لذلك فان ادخال المستخدم للحروف الكبيرة أو الصغيرة لايؤثر فى البحث عن هذه الحروف . كما أن استخدام الدالة (TRIM) يؤدى الى الغاء أى مسافات موجودة بعد الحروف التى يدخلها المستخدم . وهذا يتيح له ادخال حرف واحد مثلا دون الخوف من تأثير المسافات الثلاثة الباقية على البحث .

وهذا الجزء يحتوى على السطور التالية :

```
lookup = UPPER (TRIM(lookup))
SEEK lookup
mrecord = RECNO()
```

ويجب ملاحظة أن الأمر (SEEK) وظيفته توجيه مؤشر السجلات (Record Pointer) الى السجل المطلوب . كما يتم عن طريق السطر التالى له تخزين رقم هذا السجل فى متغير الذاكرة (mrecord) . وذلك لكى يسهل الذهاب الى هذا السجل بعد ذلك .

وفى الجزء الخامس من البرنامج يتم حساب عدد السجلات التى تحقق شرط الاسم وتخزين هذا العدد فى المتغير (howmany) . وحيث أن الملف مفهرس على حقل الاسم (name) ، لذلك يفضل استخدام (WHILE) فى البحث بدلا من (FOR) حيث أنه يقوم بتجميع السجلات التى تحقق الشرط . ولذلك يكون البحث أسرع بواسطة (WHILE) .

بعد ذلك يقوم البرنامج باختبار العدد الذى تم تخزينه فى المتغير (howmany) . فاذا كان هذا العدد صفرا فان هذا يعنى أنه لايوجد سجل يبدأ بالحروف التى أدخلها المستخدم . ولذلك تظهر الرسالة المبينة ، والتى يتم خلالها كتابة الحروف التى أدخلها المستخدم فى المتغير (lookup) باستخدام الماكرو(&) . كما يتم تحذير المستخدم عن طريق الدالة (CHR(7)) التى تؤدى الى تشغيل الجرس . وهذا الجزء يحتوى على السطور التالية :

```
IF howmany = 0
    @ 20,10 SAY "There is no & lookup"
    @ 22,10 SAY "press any key to try again"
    ? CHR(7)
    WAIT
```

```
    mrecord = 0
ENDIF(howmany = 0)
```

وفي الجزء السادس يتم دراسة حالة أخرى وهي وجود أكثر من سجل يحقق الشرط ، أى أن (howmany) أكبر من (١). في هذه الحالة يقوم البرنامج بعرض بعض بيانات هذه السجلات حتى يستطيع المستخدم تمييز السجل الذى يريد تعديله . كما يتم عرض أرقام السجلات الخاصة بهذه السجلات . ويتم سؤال المستخدم عن السجل المطلوب ليقوم بادخال رقم هذا السجل . وهذا الجزء يحتوى على السطور التالية :

```
IF howmany > 1
    CLEAR
    mrecord = 0
    SEEK lookup
    DISPLAY NAME , CLASS WHILE UPPER(NAME) = lookup
    @ ROW()+3,10 SAY "Edit which record # ?" ;
    GET mrecord PICTURE "9999"
    READ
ENDIF
```

وفي الجزء السابع يتم اختبار المتغير (mrecord) فاذا كان أكبر من صفر، فان هذا يعنى أنه تم ايجاد أحد السجلات المطابقة سواء من خلال الشرط الأول (howmany = 0) أو من خلال الشرط الثاني (howmany > 1) أو من خلال الحالة الوحيدة المتبقية وهي (howmany = 1) . لذلك يتم الذهاب الى هذا السجل باستخدام الأمر (GOTO) . ثم يتم فتح ملف التشكيل (Format file) الذى يؤدى الى عرض شاشة الادخال الخاصة بملف قاعدة البيانات المفتوح . وهذا يتيح للمستخدم ادخال التعديلات المطلوبة ، ثم يتم اغلاق ملف التشكيل مرة ثانية . وهذا الجزء يحتوى على السطور التالية :

```
IF mrecord > 0
    CLEAR
    GOTO mrecord
    SET FORMAT TO cadets
    READ
    CLOSE FORMAT
ENDIF
ENDDO (WHILE entering)
RETURN
```

الفصل الحادى عشر

برنامج مسح السجلات

يساعد هذا البرنامج على وصول المستخدم الى سجل أو عدة سجلات ومسحها . والخطوات الأولية (PSEUDOCODE) لهذا البرنامج تكون كالآتى :

١ - يتم تكوين حلقة تكرارية لمسح السجلات .
٢ - يتم السؤال عن الاسم المطلوب مسح السجل الخاص به .
٣ - عند عدم ادخال أى اسم ، يتم الرجوع الى القائمة الرئيسية .
٤ - يتم تحويل الاسم الى حروف كبيرة (Uppercase) .
٥ - يتم البحث عن الاسم المطلوب .
٦ - يتم حصر عدد السجلات التى تحتوى على نفس الاسم .
٧ - فى حالة عدم العثور على أى سجل يحتوى على هذا الاسم ، يتم تحذير المستخدم حتى يدخل اسما آخر .
٨ - فى حالة وجود عدة سجلات تحتوى على نفس الاسم ، يتم عرض هذه السجلات على المستخدم لاختيار أحدها عن طريق رقم السجل .
٩ - عند الوصول الى السجل المطلوب ، يتم سؤال المستخدم مرة ثانية للتأكد من رغبته فى مسح هذا السجل .
١٠- عند تأكد المستخدم من رغبته فى مسح السجل يتم وضع علامة على هذا السجل تمهيدا لمسحه .
١١- يتم السماح للمستخدم بادخال أسماء أخرى حتى ينتهى من تحديد الأسماء التى يريد مسحها .
١٢- يتم حصر عدد السجلات التى تم وضع علامات عليها لمسحها .
١٣- طالما كانت هناك سجلات عليها علامات المسح (Marked for deletion) ، يتم تنفيذ الآتى :

- يتم عرض بيانات السجلات التى تم وضع علامات عليها .
- يتم التأكد من رغبة المستخدم فى مسح هذه السجلات كلها مسحا دائما (Permanently) .
- اذا أراد المستخدم استرجاع بعض هذه السجلات يتم اعطاؤه الفرصة لاستعادة أحد هذه السجلات .
- عندما يتأكد المستخدم من رغبته فى مسح السجلات الباقية يتم مسح هذه السجلات مسحا دائما (Permanently) .

١٤ - تعود القائمة الخاصـــة بالمسح للظهور حتى يمكن تكرار العملية مع سجلات أخرى .
١٥ - بعد الانتهاء يتم الرجوع الى القائمة الرئيسية .

بعد كتابة الخطوات الأولية (PSEUDOCODE) ، يتم كتابة البرنامج الذى يقوم بتنفيذها كالآتى :

```
* - - - - - - Set up loop for deleting records .
entering = .T.
DO WHILE entering
    * - - - - - - - -  Ask for name of person to lookup
    CLEAR
    lookup = SPACE(4)
    @ 10,12 SAY "Enter name of person to delete"
    @ 12,12 SAY "or just press return to exit"
    GET lookup
    READ
    * - - - - - -  If no name entered , skip all
    * - - - - - -  commands between here and Enddo
    IF lookup = "  "
       entering = .F.
       LOOP
    ENDIF (lookup = "   ")

    *  - - - - - -  convert lookup to uppercase to match
    *  - - - - - -  index file
    lookup = UPPER(lookup)

    *  - - - -  Try to find requested name, and
    * - - - - - - remember record number
    SEEK lookup
    mrecord = RECNO()

    *  - - - - - - Count howmany there are
    COUNT WHILE UPPER (NAME) = lookup TO howmany
    * - - - If no record has that name , warn the user
    * - - - - - -  to try again
    IF howmany = 0
        @ 20,10 SAY "There is no & lookup"
        @ 22,10 SAY "press a key to try again"
        ? CHR(7)
        WAIT " "
        mrecord = 0
    ENDIF (howmany = 0)
```

```
* - - - - -  If more than one record has that name
* - - - - -  display records to the user.
IF homany > 1
    CLEAR
    mrecord = 0
    SEEK lookup
    LIST NAME , CLASS WHILE UPPER(NAME) = lookup ;
    @ ROW()+3,10 SAY "Delete which one?"
         GET mrecord PICTURE "9999"
    READ
ENDIF


* - - - - - -If the value of (mrecord) greater than
* - - - - - - zero double check then delete
IF mrecord > 0
    GOTO mrecord
    CLEAR
    DISPLAY NAME , CLASS
    ?
    WAIT "Delete this record? (Y/N)" TO answer

    * - If answer is yes, mark record for deletion
    IF UPPER(answer) = "Y"
        DELETE RECORD mrecord
        ENDIF(answer)
    ENDIF(mrecord > 0)
    ENDDO (while entering)

    * - Before exiting , verify deletion and pack
    COUNT FOR DELETED() TO nodels
    oktopack = "N"
    DO WHILE oktopack = "N" .AND. nodels > 0
        CLEAR
        ? "Records to be deleted .."
        ?
        DISPLAY NAME , CLASS FOR DELETED()
        @ 23,1 SAY "Delete all these? (Y/N)" ;
```

```
      GET oktopack PICTURE "!"
   READ
   IF oktopack < > "Y"

     * - if not ok to pack , recall a record
     delrec = 0
     @ 23,1 SAY "Recall which one";
       GET delrec PICTURE "9999
     READ
     * - if record number entered and record
     * - is indeed deleted, recall it
     IF delrec > 0
        GOTO delrec
        IF DELETED()
           RECALL RECORD delrec
           nodels = nodels -1
        ENDIF
     ENDIF
 ELSE

    * - - - -  if ok to pack , pack and show
    * - - - - -  the process of packing
    SET TALK ON
    PACK
    SET TALK OFF
 ENDIF (oktopack)
ENDDO(oktopack)
RETURN
```

الهدف من هذا البرنامج هو البحث عن سجل معين ومسحه . والجزء الخاص بالبحث لايختلف عن برنامج التصحيح أو برنامج التقارير . حيث يقوم البرنامج بعرض رسالة للمستخدم لادخال الاسم المطلوب البحث عنه . ثم البحث عن هذا الاسم باستخدام أوامر البحث المعروفة ثم عرض بيانات السجلات التي تشترك في هذا الاسم حتى يقوم المستخدم باختيار سجل محدد منها عن طريق رقم السجل .

ولكن المطلوب من البرنامج بعد ذلك مسح هذا السجل . وعملية المسح تتسم بشيء من الخطورة حيث أن المستخدم قد يمسح سجلا ثم يكتشف بعد ذلك أنه قد

مسح سجلا مطلوبا عن طريق الخطأ . ولذلك فان البرنامج يجب أن يتيح للمستخدم عدة مراحل من الاختبار والتحذير حتى يتأكد تماما أن هذا السجل هو السجل المطلوب مسحه .

والجزء الأول من البرنامج لايحتاج الى اعادة شرحه حيث أنه سبق شرحه فى برنامج التصحيح وبرنامج التقارير .

والجزء الجديد هو الذى يبدأ بعد الوصول الى رقم السجل المطلوب مسحه . وهو الجزء الذى يبدأ بمجموعة السطور التالية :

```
* - - - - - If the value of (mrecord) greater than zero,
* - - - - -  double check then delete
IF mrecord > 0
     GOTO mrecord
     CLEAR
     DISPLAY NAME , CLASS
     ?
     WAIT "Delete this record? (Y/N)" TO answer

     * - - If answer is yes, mark record for deletion
     IF UPPER(answer) = "Y"
          DELETE RECORD mrecord
     ENDIF(answer)
ENDIF(mrecord > 0)
ENDDO (while entering)
```

ويمكن للمستخدم من خلال الحلقة التكرارية الأولى (WHILE entering) تحديد عدة سجلات مطلوب مسحها . وفى كل مرة يتم عرض بيانات السجل عليه حتى يتأكد أن هذا هو السجل المطلوب . وهذه العملية تؤدى فى النهاية الى وضع علامات أمام عدة سجلات تمهيدا لمسحها .

وبعد ذلك يتم تكوين حلقة تكرارية أخرى تتيح للمستخدم المسح النهائى لهذه السجلات . وهذه الحلقة التكرارية تظهر فى السطور التالية :

```
* - - - - - Before exiting , verify deletion and pack
COUNT FOR DELETED() TO nodels
oktopack = "N"
```

Converted by Tiff Combine - (no stamps are applied by registered version)


```
DO WHILE oktopack = "N" .AND. nodels > 0
      CLEAR
      ? "Records to be deleted .."
      ?
      DISPLAY NAME , CLASS FOR DELETED()
      @ 23,1 SAY "Delete all these? (Y/N)";
           GET oktopack PICTURE "!"
      READ
      IF oktopack < > "Y"

      * - - - - - if not ok to pack , recall a record
      delrec = 0
      @ 23,1 SAY "Recall which one (by record#)";
          GET delrec PICTURE "9999"
      READ

      * - - - - -if record number entered and record is
      * - - - - -  indeed deleted, recall it
        IF Delrec > 0
           GOTO delrec
           IF DELETED()
                RECALL RECORD delrec
                nodels = nodels -1
           ENDIF
        ENDIF
       ELSE

        * - - - - -  if ok to pack , pack and show
        * - - - - -  the process of packing
        SET TALK ON
        PACK
        SET TALK OFF
       ENDIF (oktopack)
 ENDDO(oktopack)
 RETURN
```

وهذه المجموعة من السطور تبدأ بحساب عدد السجلات التى تم وضع علامات عليها

باستخدام الأمر (COUNT) وتخزين هذا العدد في المتغير (nodels) . ويجب ملاحظة أن الأمر (COUNT) يقوم بانشاء المتغير آليا ، أي ليست هناك حاجة لانشاء المتغير (nodels) قبل استخدامه . ثم تبدأ الحلقة التكرارية ويلاحظ قبلها انشاء المتغير (oktopack) واعطاؤه القيمة ("N") وذلك حتى يبدأ تنفيذ الحلقة التكرارية مرة واحدة بصرف النظر عن القيمة التي يدخلها المستخدم بعد ذلك في هذا المتغير . ويتوقف تنفيذ الحلقة التكرارية على شرطين . الأول هو (oktopack = "N") وهذا يكون صحيحا في البداية . والشرط الثاني هو (nodels > 0) وهذا يعني أن هناك سجلات تم وضع علامات عليها . أما اذا لم تكن هناك سجلات من هذا النوع فهذا معناه أن المستخدم لم يدخل أي اسم في البداية لمسحه ، أو أن الاسم الذي تم ادخاله لم يتم العثور عليه . وفي هذه الحالة لايتم تنفيذ الحلقة التكرارية ويعود البرنامج الى القائمة الرئيسية .

وعندما يتحقق الشرطان تبدأ الحلقة التكرارية في التنفيذ وتظهر بيانات الاسم والفرقة (CLASS) الخاصة بالسجلات التي تم وضع علامات عليها . كما يظهر سؤال للمستخدم اذا كان يريد مسح هذه السجلات كلها أم يريد استعادة أحدها . فاذا أراد استعادة أحدها فان الشرط الموجود مع الأمر ("IF oktopack <> "Y) يتحقق . وهذا يؤدي الى ظهور سؤال آخر للمستخدم عن رقم السجل الذي يريد استعادته ، مع ملاحظة أن السجلات تكون معروضة على الشاشة ومعها أرقام السجلات . وعندما يدخل المستخدم رقم السجل المطلوب استعادته ، يتم تخزين هذا الرقم في المتغير (delrec) . ونتيجة لذلك فان الشرط (IF delrec > 0) يتحقق وينتقل المؤشر الى هذا السجل . ثم يتم اختبار هذا السجل عن طريق الدالة (()DELETED) للتأكد أن السجل تم وضع علامة عليه لمسحه . وفي هذه الحالة يتم استعادة هذا السجل باستخدام الأمر (RECALL) ثم يتم انقاص عدد السجلات الموجودة في المتغير (nodels) بمقدار (١) . وتتكرر هذه العملية حتى يقوم المستخدم باستعادة جميع السجلات التي لا يريد مسحها . فاذا كان هناك سجلات باقية يراد مسحها ، فان المستخدم يجيب على السؤال الخاص بمسح جميع السجلات بكتابة (Y) وهذا يؤدي الى عدم تحقق الشرط ("IF oktopack <> "Y) وبالتالي تنفيذ الأوامر بعد (ELSE) . ومن خلال هذه الأوامر يتم اتمام عملية المسح باستخدام الأمر (PACK) . ويلاحظ أنه تم استخدام الأمر (SET TALK ON) قبل استخدام الأمر (PACK) وذلك حتى يرى المستخدم الرسالة الدالة على اتمام عملية المسح . ثم يتم الرجوع الى البرنامج الرئيسي باستخدام الأمر (RETURN) .



الجزء الثالث

نظام المخازن

* توصيف النظام
* برنامج القائمة الرئيسية
* برنامج تشغيل الملف الرئيسى
* برنامج تشغيل ملف المبيعات
* برنامج تشغيل ملف الاضافة
* برنامج تحديث البيانات



الفصل الثاني عشر

توصيف النظام



من النظم الشائعة الاستخدام فى الوقت الحالى نظم التحكم فى المخزون (Stock Control) وهي النظم التي تتيح التحكم في كمية البضائع المخزونة ومتابعة البضائع الواردة والبضائع المصروفة . واستخدام قواعد البيانات فى هذه النظم أدى الى كفاءة ملحوظة في السيطرة على هذه العملية . وقد روعى فى هذا الكتاب شرح برنامج مخازن متكامل لأنه يوفر لمخطط البرامج شرحا لمهارات متعددة فى كتابة البرامج حيث أنه يستخدم نظام التعديل المجمع (Batch Updating) من خلال ملفات الحركة (Transaction Files) . كما يمكن ربط هذا البرنامج ببرنامج حسابات العملاء (Accounts Receivable) المشروح فى الجزء الرابع لانشاء نظام متكامل .

١٢ - ١ تصميم النظام

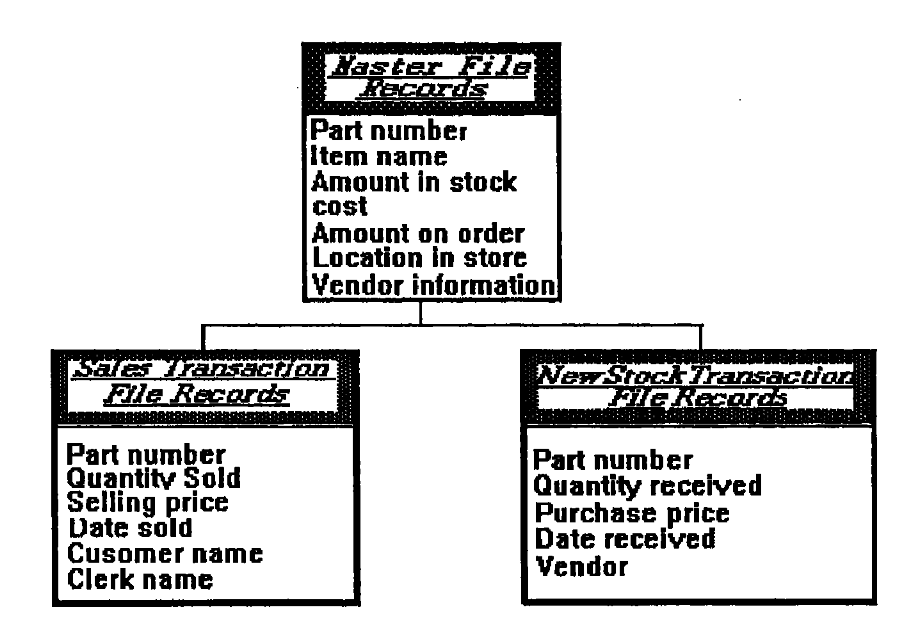
يتطلب نظام المخازن بصفة عامة تصميم عدة ملفات للبيانات أولها يسمى الملف الرئيسى (Master File) . وهذا الملف الرئيسى يشتمل على بيانات عن الأصناف الموجودة حاليا بالمخزن مثل كمية هذه الأصناف بالاضافة الى سعر كل صنف والكمية تحت الطلب من هذا الصنف (In Order) وحد الطلب له (Reorder Point) واسم البائع (Vendor) الذى يتم شراء هذا الصنف منه وعنوانه . كما يمكن أيضا أن يتضمن بيانات عن مكان كل صنف فى المخزن حتى يسهل على المستخدم الوصول اليه .

ويتطلب نظام المخازن أيضا تصميم ملفات تتضمن بيانات عن حركة الأصناف الموجودة فى المخزن ، وتسمى ملفات الحركة (Transaction Files) . وأحد هذه الملفات يشمل بيانات عن حركة بيع الأصناف مثل بيانات العميل ، ومتى تم البيع وبأى كمية ورقم فاتورة الشراء ويسمى هذا الملف ملف المبيعات (Sales File) .

وهناك ملف حركة آخر يتضمن بيانات عن الأصناف الجديدة التى يتم ادخالها الى المخزن (Newstock File) .

ويمكن ملاحظة العلاقة بين هذه الملفات الثلاثة من الشكل (١٢ - ١)

ويلاحظ من الشكل أن هناك حقلا مشتركا بين الملفات الثلاثة وهو حقل رقم الجزء (Part number) . وهذا الحقل يسمى حقل المفتاح (Key Field) الذى يتم عن طريقه ربط الملفات الثلاثة .



شكل (١٢ - ١)

١٢ - ٢ حقل المفتاح (Key Field)

كما سبق الايضاح فى الجزء الأول من الكتاب ، فعند تصميم نظام يتعامل مع عدة ملفات قواعد بيانات ، يجب تحديد حقل مشترك بين هذه الملفات حتى يمكن ربط هذه الملفات من خلاله . وهذا الحقل يجب أن يكون منفردا (Unique) لكل سجل فى الملف الرئيسي (Master File) . كما يجب أن يكون بنفس الاسم والنوع والطول فى الملفات الثلاثة .

وفي معظم نظم المعلومات يكون هناك رقم أو كود يمكن استخدامه كحقل مفتاح لأنه يكون منفردا (Unique) . ففي البنوك مثلا يستخدم رقم الحساب كحقل مفتاح حيث لايوجد عميلان يشتركان في رقم حساب واحد . وكذلك على مستوى الدولة يستخدم الرقم الشخصى (Social Security Number) كمفتاح للحصول

على بيانات أى شخص . وبالنسبة لنظام المخازن ، فان أنسب رقم يمكن استخدامه كحقل مفتاح هو رقم الجزء . حيث أن كل جزء له رقم جزء (Part Number) منفرد (Unique) .

١٢ - ٣ وظائف النظام

يجب تصميم نظام يسمح للمستخدم بمتابعة البضائع المخزونة والبضائع تحت الطلب ومكان هذه البضائع فى المخزن . وعند ادخال أى أصناف غير موجودة أصلا بالمخزن يتم تحديد رقم الجزء (Part Number) الخاص بها واضافة كمية كل صنف الى الملف الرئيسى . كمايجب أن يوفر النظام التقارير التى توضح حالة الأصناف بالمخزن وتحدد الأصناف التى يجب طلبها . كما يقوم بانشاء طلبات الشراء آليا .

كما يجب أن يسمح النظام للمستخدم بمتابعة حركة الأصناف ، من حيث الأصناف التى يتم بيعها والشخص الذى يقوم بعملية البيع والعميل الذى تباع له الأصناف ، وتاريخ البيع ورقم الفاتورة . كما يجب أن يسمح أيضا بمتابعة الأصناف الواردة لتعويض الأصناف الناقصة فى المخزن . كما يجب أن يقوم النظام آليا بتحديث البيانات الموجودة فى الملف الرئيسى (Master File) من ملف المبيعات (Sales) وملف الوارد (New Stock) .

١٢ - ٤ تحديد حقول الملفات

فى معظم الأحيان تكون أسهل وسيلة لتحديد حقول ملف قاعدة البيانات هى تحديد المخرجات المطلوبة من هذا الملف وهى التقارير المطلوب انشاؤها . فمثلا لتحديد حقول الملف الرئيسى (Master File) يتم دراسة محتويات التقارير المطلوبة من هذا الملف . فأحد هذه التقارير هو التقرير الذى يوضح موقف أو حالة الأصناف الموجودة فى المخزن . هذا التقرير يشتمل على رقم الجزء واسم الجزء والكمية الموجودة من هذا الجزء فى المخزن وسعر الشراء ومكان هذا الجزء فى المخزن والكمية تحت الطلب من هذا الجزء .

وهناك تقرير آخر يسمى تقرير اعادة الطلب (Reorder Report) يتم من خلاله عرض قائمة بالأصناف المطلوبة . كما أن هناك تقريرا آخر يسمى تقرير تحت الطلب (On Order Report) ، ويعرض قائمة بالأصناف الجارى طلبها . كما أن هناك طلبات الشراء التى يقوم المستخدم باصدارها لشراء أصناف جديدة .

ولتوفير هذه المخرجات يجب أن يشتمل الملف الرئيسى (Master File) على البيانات التالية :

Part number
Item name
Quantity in stock
Purchase price
Reorder point
Quantity on order
Location in warehouse
Vendor name
Vendor address
Date of last update
Date of last order
Quantity to order

كما يجب أن يوفر النظام أيضا تقارير توضح كل حركة بيع . وهذه التقارير تحتوى على رقم الجزء والكمية ورقم الفاتورة (Invoice Number) واسم الشخص القائم بالبيع واسم العميل وسعر البيع وتاريخ البيع . ولذلك فان ملف المبيعات (Sales File) يجب أن يحتوى على البيانات الآتية :

Part number
Invoice number
Salesperson's name
Quantity sold
Selling price
Date sold
Whether posted or not

كما أن النظام يجب أن يوفر تقارير توضح حركة الأصناف الواردة وتعرض رقم الجزء والكمية الواردة وثمن الشراء وتاريخ الورود واسم البائع (Vendor) . لذلك فان ملف الأصناف الواردة (New Stock File) يجب أن يحتوى على البيانات الآتية :

Part number
Quantity
Purchase price
Date received
Vendor name
Whether posted or not

١٢ - ٥ تصميم قاعدة البيانات

كما سبق الايضاح فان نظام المخازن في العادة يحتوى على ثلاثة ملفات قواعد بيانات . وهى الملف الرئيسي وليكن اسمه (Master.dbf) وملفين حركة أحدهما خاص بالبيعات ويسمى (Sales.dbf) والآخر خاص بالأصناف الواردة ويسمى (Newstock.dbf) .

١٢ - ٥ - ١ انشاء الملف الرئيسي (Master File)

يتم انشاء الملف الرئيسي عن طريق قوائم برنامج المساعد (Assistant) كما سبق الايضاح في الكتاب الأول . كما يمكن انشاؤه أيضا باستخدام الأمر (CREATE) وذلك كالآتي :

```
CREATE MASTER
```

وفي هذه الحالة تظهر شاشة هيكل ملف قاعدة البيانات (Structure) التي يتم عن طريقها كتابة اسم كل حقل ونوعه وطوله وعدد الأرقام العشرية ان وجدت . ويتم تكوين الملف كالآتي :

	FIELD NAME	TYPE	WIDTH	DEC
1	PART_NO	Character	5	
2	P_NAME	Character	20	
3	QTY	Numeric	7	2
4	COST	Numeric	9	2
5	REORDER	Numeric	7	2
6	ON_ORDER	Numeric	7	2
7	LOCATION	Character	5	
8	VENDOR	Character	25	
9	VENDOR_ADD	Character	25	
10	DATE	Date	8	
11	ORDER_DATE	Date	8	
12	NEW_ORDER	Numeric	7	2

ومن هـــذا الشكل يلاحظ أن الحقل رقم (١) هو الحقل الخاص برقم الجزء (Part Number) وهو حقل حرفي يتكون من خمـــسة حروف . والحقـل رقم (٢) هو حقل اسم الجزء (Part Name) وهو حقل حرفي يتكـون مـن عشريـن حرفا . والحقـل رقم (٣) هـو حقـل الكمية (Quantity) وهـو حـقـل عددى يتكون من سبعة أرقام متضمنا رقمين عشريين . والحقل رقم (٤) هـو حقل سعر الشراء (Cost) وهو حقل عددى يتكون من تسعة أرقـــام متـضمنا رقمـين عـشريين . والحقل رقم (٥) هو حقـل حـد الـطلب (Reorder Point)ويمثل الحد الذى يجب عنده اعادة طلب هـذا الـصنـف ويتكون من سبعة أرقام متضمنا رقمين عشريين . والحقل رقم (٦) هـو حـقـل الكمية الجارى طلبها ، وهو حقل عددى مكون من سبعة أرقام متضمنا رقمـين عشريين . والحقل رقم (٧) هو حقل مكان الصنف فى المخـزن (Location) وهو حقل حرفي مكـون مـن خمسة حـروف . والحقـل رقـم (٨) هـو حقـل البـائـع (Vendor) وهو حقل حرفي مكون من خمسة وعشرين حرفا . والحقل رقم (٩) هـو حقل عنوان البائع وهو حقل حرفي مكون من خمسة وعشرين حرفا . والحقل رقم (١٠) هو حقل تاريخ آخر تحديث للملف وهو حقل تاريخـي مكـون مـن ثمانـية حـــروف . والحـقـل رقم (١١) هو حقل تاريخ آخر طلب للصنـف (Date of last order) . والحـــقل رقم (١٢) هو حقل الكمية التى يجب طلبها من هذا الصنف ، وهو حقل عددى مكون من سبعة أرقام متضمنة رقمين عشريين .

ولانشاء ملف الفهرس الخاص بهذا الملف على أن يكون الحقل الخاص برقـم الجزء هـو الحقل الفهرسى (Key Field) يتم كتابة السطريـن التاليـين مـن مشيرة النقطة .

```
USE Master
INDEX ON Part_no TO Master
```

١٢ - ٥ - ٢ انشاء ملف المبيعات (Sales File)

يتم انشاء ملف المبيعات (Sales.dbf) بنفس الطريقة كما سبق انشـاء الملف الرئيسى . ويجب ملاحظة أن حقل رقم الجزء (Part_no) يجب أن يكـون موجـودا فى هذا الملف وبطول خمسة حروف مثل الملف الرئيسى تماما . ويـتـم تكوين الملف كالآتى :

FIELD	NAME	TYPE	WIDTH	DEC
1	PART_NO	Character	5	
2	INVOICE_NO	Numeric	6	0
3	CLERK	Character	12	
4	CUSTOMER	Character	12	
5	QTY	Numeric	7	2
6	PRICE	Numeric	9	2
7	DATE	Date	8	
8	POSTED	Logical	1	

والحقل رقم (١) هو حقل رقم الجزء (Part number) . والحقل رقم (٢) هو حقل رقم الفاتورة (Invoice number) وهو حقل عددى مكون من ستة أرقام . والحقل رقم (٣) هو حقل اسم الموظف القائم بالبيع وهو قد يكون اسم الموظف أو كود معين خاص به حسب الحاجة . وهو حقل حرفى مكون من (١٢) حرفا . والحقل رقم (٤) هو حقل اسم العميل (Customer) وهو حقل حرفى مكون من (١٢) حرفا . والحقل رقم (٥) هو حقل الكمية المباعة وهو حقل عددى مكون من سبعة أرقام متضمنة رقمين عشريين . والحقل رقم (٦) هو حقل سعر البيع وهو حقل عددى مكون من تسعة أرقام ورقمين عشريين . والحقل رقم (٧) هو حقل تاريخ البيع ، وهو حقل تاريخي مكون من ثمانية حروف . والحقل رقم (٨) هو حقل الترحيل (Posting) وهو حقل منطقى (Logical) يستخدم للتحكم في ترحيل بيانات الصنف الى الملف الرئيسى ، وضمان عدم تكرار ترحيلها .

ولانشاء ملف الفهرس الخاص بهذا الملف يتم كتابة السطرين التاليين من مشيرة النقطة (Dot Prompt) :

```
USE Sales
INDEX ON Part_no TO Sales
```

١٢ - ٥ - ٣ انشاء ملف الأصناف الواردة

يتم انشاء ملف الأصناف الواردة (NewStock.dbf) بنفس الطريقة كما سبق الايضاح . مع ملاحظة ضرورة وجود حقل رقم الجزء (Part Number) بنفس الاسم والطول والنوع . ويتم تكوين الملف كالآتى :

FIELD	NAME	TYPE	WIDTH	DEC
1	PART_NO	Character	5	
2	QTY	Numeric	7	2
3	COST	Numeric	9	2
4	DATE	Date	8	
5	VENDOR	Character	50	
6	POSTED	Logical	1	

والحقل رقم (١) هو حقل رقم الجزء (Part number) . والحقل رقم (٢) هو حقل الكمية الواردة وهو حقل عددى مكون من سبعة أرقام متضمنة رقمين عشريين . والحقل رقم (٣) هو حقل ثمن الشراء وهو حقل عددى مكون من تسعة أرقام متضمنة رقمين عشريين . والحقل رقم (٤) هو حقل تاريخ وصول الصنف وهو حقل تاريخي مكون من ثمانية حروف . والحقل رقم (٥) هو حقل اسم البائع (Vendor) القائم بالتوريد وهو حقل حرفي مكون من خمسين حرفا . والحقل رقم (٦) هو حقل الترحيل وهو حقل منطقى (Logical) يستخدم للتحكم فى ترحيل بيانات الصنف الى الملف الرئيسى ، وضمان عدم تكرار ترحيلها .

ولانشاء ملف الفهرس الخاص بهذا الملف يتم كتابة السطرين التاليين من مشيرة النقطة (Dot Prompt) .

```
USE Newstock
INDEX ON Part_no TO Newstock
```

وهكذا يتكون النظام حتى الآن من ستة ملفات . ثلاثة منها ملفات قواعد بيانات (Database files) . والثلاثة الآخرين ملفات فهرس (Index files) .

١٢ - ٦ تصميم البرنامج

بعد تصميم ملفات قواعد البيانات كما سبق الايضاح يأتى دور البرنامج . وهو فى الواقع ليس برنامجا واحدا ولكنه عدة برامج كما سيتم الايضاح . وهذه البرامج تهدف الى السيطرة على ثلاثة نظم مستقلة ولكنها مرتبطة فيما بينها . النظام الأول هو نظام ادارة المخزون الحالى أو الفعلى ، والذي يكون مسئولا عن الحالة الفعلية للأصناف الموجودة بالمخزن ومتابعة حالتها وتحديد الأصناف التى تصل الى حد الطلب

(Reorder Point) وهكذا . والنظام الثاني هو حركة المبيعات التي يتم ادخالها بـواسطة الموظف المختص في نقطة البيع (Point of sale) . والنظام الثالث هو حركـة الاضـافة (Newstock) التي يتم فيها تسجيل الأصناف الواردة الى المخزن بواسطة الموظف المختص .

لذلك فان البرنامج يمكن تقسيمه بصفة مبدئية الى أربعة برامج . احدها برنامج رئيسي يقوم بتشغيل البرامج الثلاثة الأخرى . والبرامج الثلاثة الأخرى تقوم بتشغيل برامج فرعية أخرى سيتم دراستها فيما بعد . ولكن سيتم التركيز في هذه المرحلة على هذه البرامج الأربعة والتي تتضح من الشكل (١٢ - ٢) .

ويلاحظ من الشكل أن البرنامج الرئيسي (IMenu.prg) هو البرنامج الذي يحتوى على القائمة الرئيسية التي يتم من خلالها تشغيل البرامج الأخرى . وبناء على اختيار المستخدم يتم التفرع الى البرنامج (MMenu.prg) الذي يشغل الملف الرئيسي للأصناف (Master File) لمتابعة الأصناف الموجودة في المخزن . أو يتم التفرع الى البرنامج (SMenu.prg) الذي يشغل ملف المبيعات (Sales File) لتابعة موقف الأصناف التي يتم بيعها . أو يتم التفرع الى البرنامج (NMenu.prg) الذي يشغل ملف الاضافة (Newstock file) لمتابعة موقف الأصناف التي يتم توريدها .

IMenu.prg

1 - Manage the Master Inventory
2 - Manage Sales Information
3 - Manage New Stock Information
4 - Exit

MMenu.prg	SMenu.prg	NMenu.prg
Add New Part Numbers	Record Sales	Record New Items
Print Reports	Print Reports	Print Reports
Make Changes	Make Changes	Make Change
Update the Master File		

شكل (١٢ - ٢)



الفصل الثالث عشر

برنامج القائمة الرئيسية



برنامج القائمة الرئيسية (IMenu.prg) هو البرنامج الذي يقوم بتشغيل القائمة الرئيسية التي يختار المستخدم من خلالها أحد النظم الثلاثة كما سبق الايضاح . وهو لايختلف في تركيبه عن أي برنامج رئيسي يتم من خلاله عرض قائمة اختيارات (Menu choices) . وكالعادة قبل كتابة أي برنامج يفضل كتابة الخطوات الأولية (PSEUDOCODE) بأي لغة يجيدها مخطط البرامج . حتى يمكن بعد ذلك كتابة الأوامر بلغة برامج عائلة (DBase) التي تنفذ هذه الخطوات .

١٣ - ١ كتابة الخطوات الأولية (PSEUDOCODE)

يقوم البرنامج الرئيسي في البداية بانشاء متغير ذاكرة تاريخي اسمه (T_date) ويتم تخزين تاريخ اليوم الحالى ((DATE() في هذا المتغير . ثم يتم عرض هذا التاريخ وسؤال المستخدم اذا كان هذا هو التاريخ السليم أم لا . وفي حالة اختلافه يقوم المستخدم بتعديل التاريخ . وهذه الخطوة مهمة جدا لأن هذا المتغير (T_date) سيتم استخدامه في مواضع متفرقة داخل البرنامج . ثم يقوم البرنامج بسؤال المستخدم عن الاختيار المطلوب من القائمة التي تظهر أمامه . وبناء على اختيار المستخدم يتم التفرع الى البرنامج الذي يحقق هذا الاختيار .

والخطوات الأولية للبرنامج (PSEUDOCODE) يتم كتابتها كالآتي :

١ - مسح كل متغيرات الذاكرة
٢ - تجهيز بيئة البرنامج (environment)
٣ - مسح الشاشة
٤ - انشاء متغير ذاكرة لتاريخ اليوم الحالى
٥ - تكوين حلقة تكرارية لعرض القائمة الرئيسية للبرنامج
٦ - مسح الشاشة وعرض القائمة الرئيسية
٧ - استقبال اختيار المستخدم
٨ - التفرع الى البرنامج المطلوب
٩ - استمرار الحلقة التكرارية حتى يختار المستخدم الخروج
١٠- الخروج من برنامج (DBase III+)

١٣ - ٢ كتابة البرنامج

يتم كتابة البرنامج كما سبق الايضاح بكتابة (MODIFY COMMAND) يليه اسم البرنامج وهو (IMenu) . وغير مطلوب اضافة الامتداد في هذه الحالة لأن البرنامج يضيف الامتداد (.prg) آليا . و البرنامج يتيح استخدام الحروف الأربعة الأولى فقط من كل أمر . فمثلا يمكن كتابة (MODI COMM) وهذا يوفر وقتا كبيرا

عند تعديل البرنامج عدة مرات . ويتم كتابة أوامر البرنامج كالآتي :

```
*********************************************** IMenu.prg
*    Main menu for the inventory system
*
SET STATUS OFF
CLEAR ALL
SET TALK OFF
SET BELL OFF
SET SAFETY OFF
SET HEADING OFF
CLEAR

* - - - - - - -  Create underline variable Uline
Uline = REPLICATE ("_" , 80)

* - - - - - - -  Create memory variable for today's date
T_Date = DATE()
@ 17,5 SAY "To change date type new date and press" +;
    "Return "
@ 15,5 SAY "Today's date =" GET T_Date PICT "99/99/99"
READ

* - - - - - - -  Set up loop for presenting main menu.
IChoice = 0
DO WHILE IChoice # 4
    CLEAR
    @ 2,1 SAY 'Inventory system main menu'
    @ 2,60 SAY DTOC(T_Date) + "    " + TIME()
    @ 3,0 SAY Uline
    ?
    ?
    TEXT
        1. Manage master inventory
        2. Record sales
        3. Record new stock
        4. Exit
    ENDTEXT
```

```
    * - - - - -  Wait for answer
    @ 24,1 SAY "Enter choice : " GET IChoice PICT "9";
    RANGE 1,4
    READ
    DO CASE
        CASE IChoice = 1
            DO MMenu
        CASE IChoice = 2
            DO SMenu
        CASE IChoice = 3
            DO NMenu
    ENDCASE
ENDDO(while IChoice # 4)
* - - - - -  When done , exit
CLEAR
QUIT
```

والبرنامج يبدأ كالعادة بكتابة اسم البرنامج ووظيفته . ثم يتم تجهيز بيئة البرنامج عن طريق مجموعة من أوامر (SET) وهي كالآتي :

```
SET STATUS OFF
CLEAR ALL
SET TALK OFF
SET BELL OFF
SET SAFETY OFF
SET HEADING OFF
CLEAR
```

وللتعرف على وظيفة كل من هذه الأوامر يمكن الرجوع الى الكتاب الثاني .

والجزء الثاني من البرنامج يبدأ بانشاء متغير ذاكرة (Uline) يحتوى على حرف الشرطة السفلية (_) مكررا ثمانين مرة . وهذا يؤدى الى تخزين سطر بعرض الشاشة يمكن عرضه في أي مكان بعد ذلك . مع ملاحظة أن انشاء هذا المتغير في البرنامج الرئيسي يجعله عاما (Public) بالنسبة للبرامج الفرعية الأخرى . أي يمكن استخدامه مباشرة في أي برنامج فرعي . وهذا الأمر يظهر في البرنامج كالآتي :

```
Uline = REPLEICTE ("_" , 80 )
```

والجزء الثالث من البرنامج يبدأ بانشاء متغير الذاكرة التاريخى (T_DATE) وتخزين تاريخ اليوم الحالى فيه ، وهو التاريخ الذى يتم ادخاله عند تشغيل الجهاز من خلال نظام التشغيل (MS-DOS) . واذا لم يكن المستخدم قد أدخل التاريخ فى بداية تشغيل الجهاز ، فان البرنامج يعرض له التاريخ السابق تخزينه فى المتغير (T_Date) . ويستطيع فى هذه الحالة تعديل هذا التاريخ ليوافق تاريخ اليوم الحالى . وهذه العملية تتم من خلال مجموعة السطور التالية :

```
T_Date = DATE()
@ 17,5 SAY "To change date type new date and" + ;
"press Return "
@ 15,5 SAY "Today's date =" GET T_Date PICT "99/99/99"
```

والمتغير (T_date) مهم جدا لأنه سوف يستخدم فى ملء بيانات الحقول التاريخية فى ملفات المبيعات والاضافة آليا .

والجزء الرابع يقوم بتكوين الحلقة التكرارية لمسح الشاشة وعرض العنوان متضمنا التاريخ والوقت ثم خط أسفل هذا العنوان ثم عرض القائمة الرئيسية السابق شرحها ، ثم يتم عرض سؤال للمستخدم عن الاختيار المطلوب واستقبال هذا الاختيار فى المتغير (ICoice) . وهذا يتم من خلال السطور التالية :

```
IChoice = 0
DO WHILE IChoice # 4
    CLEAR
    @ 2,1 SAY "Inventory system main menu"
    @ 2,60 SAY DTOC(T_Date) + "     " + TIME()
    @ 3,0 SAY Uline
    ?
    ?
    TEXT
        1. Manage master inventory
        2. Record sales
        3. Record new stock
        4. Exit
    ENDTEXT
    * - - - - -  Wait for answer
    @ 24,1 SAY "Enter choice:" GET IChoice PICT "9" ;
        RANGE 1,4
    READ
```

والجزء الخامس من البرنامج يستعمل الأمر (DO CASE) في التفرع الى البرنامج المطلوب حسب اختيار المستخدم . وهذا يتم من خلال السطور التالية :

```
DO CASE
    CASE IChoice = 1
        DO MMENU
    CASE IChoice = 2
        DO SMenu
    CASE IChoice = 3
        DO NMenu
ENDCASE
```

وفي الجزء الأخير من البرنامج يتم انهاء الحلقة التكرارية ومسح الشاشة والرجوع الى نظام التشغيل . وذلك كالآتي :

```
ENDDO(while IChoice # 4)
* - - - - -  When done , exit
CLEAR
QUIT
```

١٢ - ٣ اختبار البرنامج

بعد الانتهاء من كتابة البرنامج يتم اختباره وذلك بتشغيله كالآتي :

```
DO IMenu
```

ثم ملاحظة تنفيذ خطوات البرنامج وظهور الاختيارات على الشاشة . ويمكن ادخال قيم خارج المدى المسموح وهو من (1) الى (4) وذلك بادخال (5) مثلا أو أي رقم آخر أكبر من (4) ، وملاحظة ما يحدث . وبعد انتهاء اختبار البرنامج يتم العودة الى نظام التشغيل عن طريق الاختيار (4) .

والشكل (١٢ - ١) يوضح شكل الشاشة التي تظهر عند تنفيذ هذا البرنامج .

```
Inventory system main menu     02/20/90     12:40:50

          1. Manage master inventory
          2. Record sales
          3. Record new stock
          4. Exit

   Enter choice
```

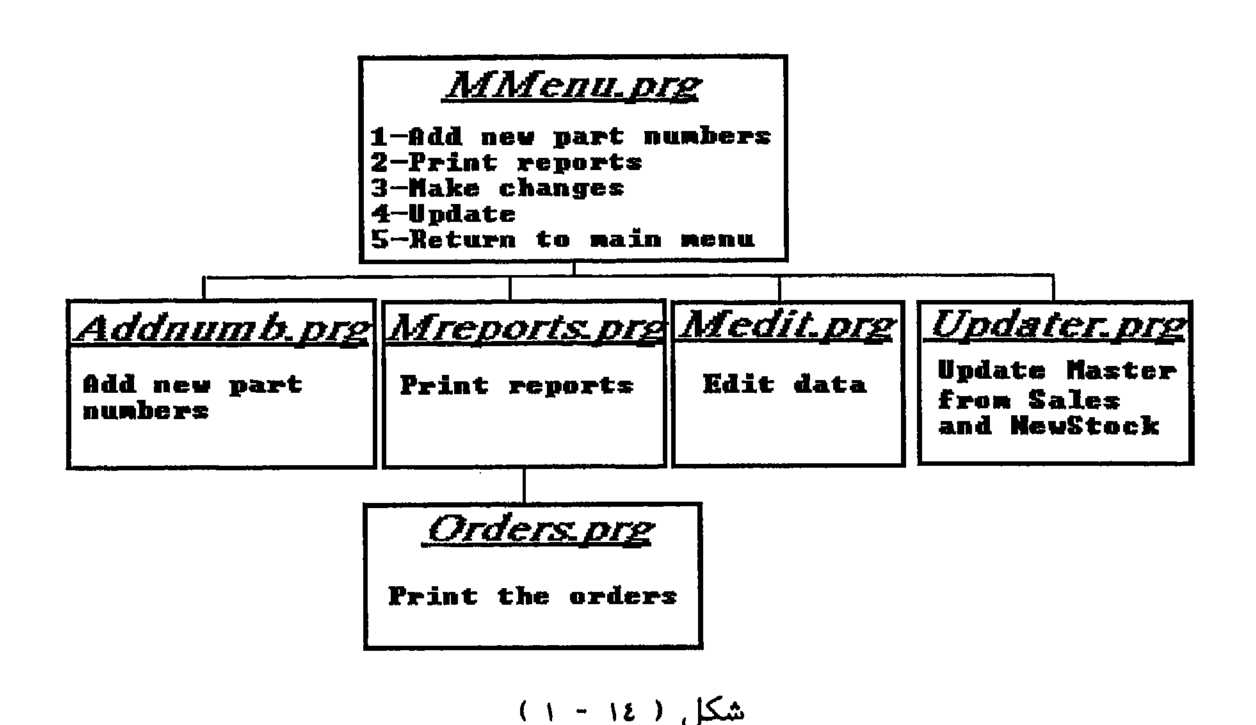
شكل (١٢ - ١)

الفصل الرابع عشر

برنامج تشغيل الملف الرئيسى



هذا البرنامج يقوم بادارة وتشغيل المعلومـــات المخـزنة فى الملف الرئيسى (Master file) بهدف تحديد الحالة الفعلية (Current status) للأصناف فى المخزن . وهذا الملف سبق تكوينه وتسميته (Master.dbf) . كما سبق انشاء الفهرس الخاص به بناء على حقل رقم الجزء (Part_no). و يتم تشغيل هذا البرنامج عند اختيار الرقم (١) من القائمـــة الرئيســية .أنظر الشكل (١٤ - ١)



شكل (١٤ - ١)

١٤ - ١ تصميم برنامج تشغيل الملف الرئيسى

يتكون برنامج تشغيل الملف الرئيسى من ستة برامج منفصلة . أحدها يمثل البرنامج الرئيسى (Main Program) الذى يتحكم فى البرامج الخمسة الأخرى أنظر الشكل (١٤ - ١) .

ويوضح الشكل اسم كل برنامج ثم وظيفة هذا البرنامج ، مع ملاحظة أن البرنامج الرئيسى (MMenu.prg) وظيفته الرئيسية عرض القائمة المبينة والتفرع الى كل برنامج من البرامج الفرعية بناء على اختيار المستخدم .

١٤ - ٢ تصميم البرنامج الرئيسى

البرنامج الرئيسى فى هذه الحالة لايختلف عن برامج القائمة الرئيسية السابق شرحها . لذلك ليست هناك حاجة لشرحه بالتفصيل . ونكتفى هنا بكتابة البرنامج كالآتى :

```
* * * * * * * * * * * * * * * MMenu.prg
*    Menu for master program of inventory system
*    Called from inventory system main menu
* - - - - -  Set up loop for presenting menu
Mchoice = 0
DO WHILE Mchoice # 5      CLEAR
    @ 2,1  SAY "Manage Master Inventory
    @ 2,60 SAY DTOC(T_Date) + "    " + TIME()
    @ 3,0  SAY Uline
    ?
    ?
    TEXT

        1. Add new part numbers
        2. Print reports
        3. Make changes
        4. Update from Sales and Newstock
        5. Return

    ENDTEXT
    @ 24,1 SAY "Enter choice(1-5)" ;
       GET Mchoice PICT "9" RANGE 1,5
    READ
    DO CASE
        CASE Mchoice = 1
            DO Addnumbs
        CASE Mchoice = 2
            DO Mreports
        CASE Mchoice = 3
            DO Medit
        CASE Mchoice = 4
```

```
            DO Updater
    ENDCASE
ENDDO(While Mchoice # 5)

* - - - - -  when done , return to main menu.
RETURN
```

وعند تنفيذ هذا البرنامج تظهر الشاشة الموضحة فى الشكل (١٤ - ٢)

```
Manage Master Inventory      02/20/90        12:40:50
---------------------------------------------------------

          1. Add new part numbers
          2. Print reports
          3. Make changes
          4. Update from sales and Newstock
          5. Return to main menu

Enter choice (1 - 5)
```

شكل (١٤ - ٢)

١٤ - ٣ برنامج اضافة الأصناف

هذا البرنامج هو البرنامج الذى يتم التفرع اليه عند اختيار المستخدم للرقم (١) فى القائمة السابقة . فى هذه الحالة يتم تنفيذ البرنامج (Addnumbs.prg) . وقبل كتابة البرنامج يجب أولا انشاء شاشة الادخال التى سوف تستخدم فى ادخال بيانات الصنف الجديد .

١٤ - ٣ - ١ انشاء شاشة الادخال

يتم انشاء شاشة الادخال باحدى طريقتين ، الطريقة الأولى باستخدام راسم الشاشة (Screen Painter) الذي يظهر عن طريق قوائم المساعد (Assistant) أو عن طريق كتابة الأمر (CREATE SCREEN) من خلال مشيرة النقطة (Dot Prompt). والطريقة الثانية عن طريق كتابة ملف أوامر (Command File). وسنستخدم الطريقة الأولى في هذا البرنامج .

ولتنفيذ ذلك يتم كتابة السطر التالى عند مشيرة النقطة : (Dot Prompt)

```
CREATE SCREEN Iscreen1
```

وعند ظهور عمود الاختيارات على الشاشة يتم اختيار (Database File). ومن خلاله يتم اختيار ملف قاعدة البيانات (Master.dbf) ، ثم يتم اختيار (Load Fields) . حيث يتم اختيار الحقول المطلوب ظهورها في الشاشة . ويتم اختيار كل حقل عن طريق تحريك العمود الضوئى الى هذا الحقل والضغط على مفتاح الادخال . في هذه الحالة يظهر مثلث (◁) أمام اسم الحقل . ويمكن اختيار كل الحقول ما عدا حقل تاريخ آخر طلب للصنف (Order_Date) وحقل الكمية التي يجب طلبها من الصنف (New_Order) لأن هذين الحقلين يتم ملؤهما آليا من خلال البرنامج . وعند الانتهاء من ادخال الحقول يتم الضغط على مفتاح (-->) للخروج من هذه القائمة ولعرض السبورة (Blackboard) التى سيتم من خلالها تحديد مواضع الحقول حسب الحاجة .

وعند الانتهاء من تحديد مواضع الحقول على الشاشة يتم تحريك المؤشر الى العمود الضوئي الخاص برقم الجزء (Part number) ويتم تحويل حالة هذا الحقل من (Edit Get) الى (Display Say) . وذلك لكي يصبح هذا الحقل غير قابل للتعديل بواسطة المستخدم .

ولتنفيذ ذلك يتم الضغط على مفتاح (F10) والضغط على مفتاح الادخال عند الاختيار (ACTION) لتحويله الى (DISPALY SAY) . ويلاحظ في هذه الحالة اختفاء العمود الضوئى الخاص برقم الجزء من الشاشة .

ولمزيد من التفاصيل عن تصميم شاشة الادخال يمكن الرجوع الى الكتاب

الأول . والشكل رقم (١٤ - ٢) يوضح تصميم مقترح لشاشة الادخال . ويمكن تصميم أى شكل آخر حسب الحاجة .

Master Inventory File	
Part number :	Date :
Part name :	Unit cost :
Quantity in stock :	Reorder point :
Quantity on order :	
Storage location :	
Vendor	Name Address

شكل (١٤ - ٢)

١٤ - ٣ - ٢ كتابة الخطوات الأولية (PSEUDOCODE)

بعـــد تصميم الشاشة يتم كتابة الخطوات الأولية لبرنـــامج (Addnumbs.prg) وذلك كالآتي :

١- يتم فتح الملف الرئيسي (Master.dbf) وملف الفهرس الخاص به .
٢- يتم تكوين حلقة تكرارية لاضافة الأصناف الجديدة .
٣- يتم مسح الشاشة .
٤- يتم سؤال المستخدم عن رقم الصنف المطلوب اضافته .
٥- يتم اختبار هذا الرقم للتأكد أنه لم يسبق ادخاله في الملف .
٦- اذا لم يتم ادخال رقم الصنف يتم الرجوع الى القائمة الرئيسية .
٧- اذا كان الرقم موجودا يتم تنبيه المستخدم لاعادة المحاولة مرة ثانية .
٨- اذا كان الرقم غير موجود يتم اضافته وفتح شاشة الادخال لادخال بيانات هذا الصنف الجديد .
٩- يتم استمــــرار تنفيذ الحلقة التكرارية حتى يضغط المستخدم على مفتاح

الادخال دون ادخال رقم صنف جديد .
١٠- يتم الرجوع الى القائمة الرئيسية .

١٤ - ٢ - ٣ كتابة برنامج الاضافة (Addnumbs.prg)

يتم كتابة الأمر (MODI COMM Addnumbs) من مشيرة النقطة ثم يتم كتابة سطور البرنامج كالآتى :

```
* * * * * * * * * * * * * * *Addnumbs.prg
*    Add new items to the Master file
*    Called from MMenu.prg

USE Master INDEX Master

* - Set up loop for adding new part numbers
Partnumb = "X"
DO WHILE Partnumb # " "
     CLEAR
     @ 2, 1 SAY "Add New Part Numbers"
     @ 2,60 SAY DTOC(T_Date) + "   " + TIME()
     @ 3, 0 SAY Uline
     ?
     ?
     * - - - - - -  GET proposed part number
     Partnumb = SPACE(6)
     @ 15,6 SAY "Enter part number(or press " + ;
    " Return to exit)" GET Partnumb
    READ

    * -  Check to see if part number already exists
    partnumb = UPPER(partnumb)
    SEEK partnumb
    DO CASE
        CASE Partnumb = "  "
        CLEAR
    CASE FOUND()
        @ 20,10 SAY Partnumb + "already exists"
```

```
        ? CHR(7)
         WAIT 'Press any key to try again'
    CASE .NOT. FOUND()
         APPEND BLANK
         REPLACE Part_no WITH Partnumb
         REPLACE Date WITH T_Date
         SET FORMAT TO Iscreen1
         READ
         SET FORMAT TO
    ENDCASE
ENDDO(While partnumb = "   " )
* - - - - - -  Return to Master menu
RETURN
```

والجزء الأول من البرنامج يبدأ كالعادة بكتابة اسم البرنامج ووظيفته ثم تحديد البرنامج القائم باستدعائه . بعد ذلك يتم فتح ملف قاعدة البيانات (Master.dbf) وملف الفهرس المقابل (Master.ndx).

والجزء الثاني من البرنامج يتم من خلاله تكوين الحلقة التكرارية بعد انشاء المتغير الحرفى (Partnumb) واعطائه القيمة (X). والقيمة (X) فى هذه الحالة تضمن تنفيذ الحلقة التكرارية على الأقل مرة واحدة . ثم يبدأ تنفيذ الحلقة التكرارية بمسح الشاشة وعرض عنوان (Header) مع عرض التاريخ والوقت ثم يتم سؤال المستخدم عن رقم الجزء المطلوب اضافته وتخزين هذا الرقم فى المتغير (Partnumb). ويجب ملاحظة أن الضغط على مفتاح الادخال دون كتابة رقم الجزء يؤدى الى الخروج من الحلقة التكرارية والرجوع الى القائمة الرئيسية . وهذا الجزء يتكون من السطور التالية :

```
Partnumb = "X"
DO WHILE Partnumb # " "
    CLEAR
    @ 2, 1 SAY "Add New Part Numbers"
    @ 2,60 SAY DTOC(T_Date) + "   " + TIME()
    @ 3, 0 SAY Uline
    ?
    ?
    * - - - - - -  GET proposed part number
    Partnumb = SPACE(6)
    @ 15,6 SAY "Enter partnumber(or press " + ;
```

```
" Return to exit)" GET Partnumb
READ
```

والجزء الثالث من البرنامج يبدأ بتحويل رقم الجزء الى حروف كبيرة (Uppercase) ثم البحث عن رقم الجزء فى الملف الرئيسي . وحيث أن الملف مفهرس ، لذلك يستخدم الأمر (SEEK) فى البحث عن هذا الرقم .

ويحتوى هذا الجزء على السطور التالية :

```
Partnumb = UPPER(partnumb)
SEEK Partnumb
```

والجزء الرابع من البرنامج يختص باتخاذ القرار بناء على ما يدخله المستخدم . وفى الحالة الأولى عندما يكون (" " = Partumb) ، فان هذا يعنى أن المستخدم يضغط على مفتاح الادخال دون كتابة أى رقم جزء (Partnumb) . وفى هذه الحالة يتم مسح الشاشة ثم تنفيذ الأمر الذى يلى الأمر (ENDCASE) . وهذا يؤدى الى الخروج من الحلقة التكرارية لأن الشرط الموجود فى أول الحلقة التكرارية لايتحقق .

وفى الحالة الثانية (CASE FOUND()) ، فان هذا يعنى أن هذا الجزء قد سبق ادخاله فى الملف . لذلك يتم تنبيه المستخدم بعرض رسالة توضح له أن هذا الرقم موجود . وذلك بالاضافة الى تشغيل الجرس للتنبيه مع استخدام الأمر (WAIT) في عرض الرسالة الموضحة وانتظار ضغط المستخدم على أى مفتاح حتى يمكنه ادخال رقم جديد .

وفى الحالة الثالثة (CASE .NOT. FOUND()) ، فان هذا يعنى أن المستخدم أدخل رقما غير موجود داخل الملف . وفى هذه الحالة فان البرنامج يعرض شاشة الادخال ثم يسمح للمستخدم بادخال باقى بيانات الصنف مع عدم السماح له بكتابة رقم الجزء من خلال شاشة الادخال . وانما يتم ادخال رقم الجزء بواسطة البرنامج عن طريق الأمر (REPLACE) . كما يتم ادخال تاريخ آخر تعديل (Date) عن طريق البرنامج أيضا .

ويتكون هذا الجزء من السطور التالية :

```
DO CASE
    CASE Partnumb = "   "
         CLEAR
    CASE FOUND()
```

```
        @ 20,10 SAY Partnumb + "already exists"
        ? CHR(7)
        WAIT 'Press any key to try again"
    CASE .NOT. FOUND()
        APPEND BLANK
        REPLACE Part_no WITH Partnumb
        REPLACE Date WITH T_Date
        SET FORMAT TO Iscreen1
        READ
        SET FORMAT TO
    ENDCASE
```

والجزء الأخير من البرنامج يحتوى على أمر انهاء الحلقة التكرارية والعودة الى القائمة الرئيسية الخاصة بالملف الرئيسى (MMenu) . واذا لم يرد المستخدم الخروج فان الحلقة التكرارية تستمر . أما اذا أراد الخروج فانه يضغط على مفتاح الادخال دون ادخال أى رقم ، وفى هذه الحالة يتم الرجوع الى القائمة الرئيسية .

١٤ - ٤ برنامج تقارير الملف الرئيسى

يمكن تصميم تقارير الملف الرئيسى (Master.dbf) عن طريق قوائم برنامج المساعد (Assistant) أو باستخدام الأمر (CREATE REPORT) . وفى الحالتين تظهر اختيارات تصميم التقرير التى يتم عن طريقها ادخال المعاملات المختلفة التى توضح عنوان الصفحة (Page Title) وأسماء الحقول بالاضافة الى المعاملات الأخرى . وكما سبق الايضاح فان هناك طريقة أخرى لتصميم التقرير عن طريق ملف الأوامر (Command file) . وهى تتيح امكانيات أكبر فى تحديد أماكن الحقول وشكل التقرير بصفة عامة . وبالنسبة للملف الرئيسى (Master.dbf) فسوف نتعرض للطريقتين فى انشاء التقارير الخاصة به .

وهناك أربعة تقارير مطلوبة للملف الرئيسى يتم شرحها فى الأجزاء التالية :

١٤ - ٤ - ١ تقرير المخزون الحالى (Current Stock)

هذا التقرير يوضح الموقف الحالى للأصناف فى المخزن من حيث اسم كل صنف والكمية الفعلية الموجودة منه ومكانه فى المخزن وسعر الوحدة وهكذا . ويتم انشاء هذا التقرير عن طريق كتابة السطرين التاليين من مشيرة النقطة (Dot Prompt) .

```
USE Master
CREATE REPORT Allmast
```

حيث (Allmast) هو اسم ملف التقرير . ويلاحظ فى هذه الحالة ظهور قوائم تصميم التقرير السابق شرحها فى الجزء الأول ولن نتعرض لها هنا بالتفصيل ولكن سيتم توضيح المعاملات التى يجب ادخالها حتى نحصل على شكل التقرير المطلوب .والشكل التالى يوضح فيم المعاملات الخاصة بصفحة التقرير .

Pagetitle	Current stock
Page width	74
Left margin	1
Right margin	0
Lines per page	52
Double space report	NO

شكل (١٤ - ٤)

كما يتم تحديد محتويات أعمدة التقرير(Columns) كالآتى :

Column No.	Contents	Heading	Width	Decimals	Total
1	Part_no	Part No	6		
2	Title	Part Name	20		
3	Qty	On Hand	7	2	N
4	Cost	Unit Cost	9	2	N
5	Reorder	Reorder	8	0	N
6	Location	Location	9		
7	Date	Last Update	9		

شكل (١٤ - ٥)

ولتوضيح شكل التقرير بعد طباعته نفرض بيانات بعض الأصناف، ثم نكتب الأمر (REPORT FORM Allmast) . وفى هذه الحالة يظهر التقرير كالآتى مثلا :

Page No 102/20/90				Current Stock		
Part No.	Part Name	On Hand	Unit Cost	Reorder	Location	Last Update
BBB	Floppy Disk		40	14	10-A-111	02/15/90
AAA	Printer		10	800	17-B-233	02/10/90
ZZZ	Bicycle		10	80	19-C-175	01/01/90

شكل (١٤ - ٦)

١٤ - ٤ - ٢ تقرير حد الطلب (Reorder)

وهو التقرير الذى يعرض بيانات الأصناف التى تقل عن حد الطلب . ويتم انشاؤه عن طريق الأمر (CREATE REPORT Reorders) كما سبق الايضاح . ويتم استخدام المعاملات الموضحة فى الشكل التالى :

Page Title	Goods to be Reordered
Page width	77
Left margin	1
Right margin	0
Lines per page	58
Double space report	NO

شكل (١٤ - ٧)

كما يتم ادخال محتويات أعمدة التقرير (Columns) كالآتى :

Column No	Contents	Heading	Width	Decimals	Total?
1	Part_no	Part No	6		
2	Title	Part Name	20		
3	Qty	On Hand	7	2	N
4	Cost	Unit Cost	9	2	N
5	Reorder	Reorder	8	0	N
6	Location	Location	9		
7	Date	Last Update	8		

شكل (١٤ - ٨)

ولتـــوضيح شكل التقـــرير عند طباعـــته يتـــم كتـــابة الأمـــــــر (REPORT FORM Reorders) فى هذه الحالة يظهر التقرير الآتى :

Page No. 1
02/20/90

Goods to be Reordered

Part No	Part name	On Hand	On Order	reorder	Vendor Name
AAA	Printer	5	5	10	Micronet Company

شكل (١٤ - ٩)

١٤ - ٤ - ٣ تقرير الأصناف تحت الطلب

ويتم انشاء هذا التقرير بنفس الطريقة مثل التقارير السابقة عن طريق الأمر (CREATE REPORT Onorder) ، مع ادخال المعاملات الواضحة فى

الشكل التالى :

Page title	Items Currently On Order
Page Width	77
Left Margin	1
Right Margin	0
Lines per Page	58
Double space reprot	NO

شكل (١٤ - ١٠)

كما يتم ادخال بيانات الأعمدة بالاستعانة بالبيانات الموجودة فى الشكل التالى :

Column No.	Contents	Heading	Width	Decimals	Total?
1	Order_date	Order Date	8		
2	Part_no	Part No	6		
3	Title	Part Name	15		
4	On_order	On Order	7	2	N
5	Cost	Unit Cost	9	2	N
6	Cost*On_order	Total Cost	9	2	Y
7	Vendor	Vendor Name	25		

شكل (١٤ - ١١)

ولعرض التقرير يتم كتابة الأمر (REPORT FORM Onorder) .

وفي هذه الحالة يظهر التقرير كالآتى مثلا :

Page No. 1 02/22/90			Item Quantity On Order			
Order Date	Part No	Part Name	On order	Unit Cost	Total Cost	Vendor Name
01/02/90	AAA	Printer	5	800	4000	Micronet Company
TOTAL					4000	

شكل (١٤ - ١٢)

ويلاحظ هنا أن التقرير يقوم بتجميع البيانات الموجودة فى عمود التكلفة الكلية (Total Cost) ، وذلك لأنه قد سبق ادخال الاختيار(Y) فى العمود (Total?) كما هو واضح من الشكل الخاص ببيانات الأعمدة .

١٤ - ٤ - ٤ طلب الشراء (Purchase Order)

طلب الشراء هو أحد التقارير التى يتم الحصول عليها من خلال برنامج تشغيل الملف الرئيسى (Master.dbf) . والبرنامج يعرض على المستخدم بيانات الأصناف التى تصل الى حد الطلب ويتيح للمستخدم طلب العدد الذى يريده من هذه الأصناف . حيث يعرض البرنامج الشاشة التالية لكل صنف من الأصناف التى تصل الى حد الطلب .

Part number	D-100 Shirt
On hand	30
On order	10
Reorder	50
Unit cost	100
Order how many?	

شكل (١٤ - ١٢)

ومن خلال هذه الشاشة يستطيع المستخدم أن يلاحظ بسرعة عدد البدل (Suits) الموجودة فى المخزن وعدد البدل الجارى طلبها (On Order) وحد الطلب (Reorder). وبناء على ذلك يحدد عدد البدل المطلوب شراؤها لتعويض النقص الموجود . ويقوم بادخال هذا العدد أمام السؤال المبين . ويتم تكرار هذه العملية مع باقى الأصناف التى وصلت الى حد الطلب (Reorder) . وبعد الانتهاء من ادخال الأعداد المطلوبة من كل صنف يقوم البرنامج آليا بطباعة طلبات الشراء لهذه الأصناف . والشكل التالى يوضح نموذجا لأحد هذه الطلبات .

Omar Afandy Company
Abbasia street

Please send us the following items...

10	suits	100	1000
5	Printers	800	4000
Total Cost			5000

Mail to AAA Company
12 – Ahram Street

شكل (١٤ - ١٤)

١٤ - ٤ - ٥ تصميم برنامج التقارير

يتكون برنامج التقارير من برنامجين أحدهما هو البرنامج (MReports.prg) الذى يعرض القائمة الرئيسية للتقارير ، وهى القائمة التى يختار المستخدم منها التقرير المطلوب طباعته . والبرنامج الآخر هو برنامج طلبات الشراء (Orders.prg) الذى يتم عن طريقه طباعة طلبات الشراء للأصناف التى تقل عن حد الطلب فى المخزن .

١٤ - ٤ - ٦ تصميم برنامج القائمة

وهذا البرنامج لايختلف عن برامج القوائم الأخرى . ولكن يجب ملاحظة أن فتح ملف قاعدة البيانات وملف الفهرس المرتبط به يتم فى بداية البرنامج وقبل الدخول فى الحلقة التكرارية . وذلك لأن الملفين يستخدمان فى جميع التقارير بعد ذلك . والبرنامج يتكون من السطور التالية :

```
* * * * * * * * * * * * * * * *   MRoports.prg
*    Present report options for Master file
*    Called from Master menu , MMenu.prg
USE Master INDEX Master
* - - - - -  Set up loop for presenting menu
Repchoice = 0
DO WHILE Repchoice # 5
    CLEAR
    @ 2,1  SAY "Master Inventory Report Options"
    @ 2,60 SAY DTOC(T_Date) + "    " + TIME()
    @ 3,0  SAY Uline
    ?
    ?
    TEXT

            1. Entire inventory
            2. Reorder report
            3. On Order report
            4. Purchase orders
            5. Return to master menu

    ENDTEXT
    @ 24,1 SAY "Enter choice(1-5)" ;
       GET Repchoice PICT "9" RANGE 1,5
    READ
    * - - If not choosing Purchase orders, ask about
    * - - -   printer .
    CLEAR
    STORE " " TO YN, Printer
    IF Repchoice < 4
```

```
        @ 5,5 SAY "Send report to printer ?" ;
           GET YN  PICT "!"
        READ
        CLEAR

        * - - - - -  Set up for printer.
        IF YN = "Y"
            Printer = "TO PRINT"
        ENDIF
    ENDIF(Repchoice < 4)
    DO CASE
        CASE Repchoice = 1
            REPORT FORM Allmast & Printer
        CASE Repchoice = 2
            REPORT FORM Reorders FOR (QTY + On_order);
              < = Reorder & Printer
        CASE Repchoice = 3
            REPORT FORM Onorder FOR On_order > 0 ;
              & Printer
        CASE Repchoice = 4
            DO Orders
    ENDCASE

    * - - -  IF report not going to printer and not
    * - - -  exiting program , pause .
    IF YN # "Y" .AND. Repchoice # 5
    ?
    ?
    WAIT "Press any key to return to the reports menu"
    ENDIF
ENDDO (Repchoice # 5)
RETURN
```

والجزء الأول من البرنامج يبدأ بالسطور المعتادة لكتابة اسم البرنامج الذى يقوم باستدعائه . ثم يتم فتح الملف الرئيسى (Master.dbf) وملف الفهرس الخاص به (Master.ndx) . ويتم بعد ذلك تكوين الحلقة التكرارية التي يتم من خلالها عرض قائمة الاختيارات التى يختار المستخدم منها نوع التقرير المطلوب . ويتكون هذا الجزء من السطور التالية :

منها نوع التقرير المطلوب . ويتكون هذا الجزء من السطور التالية :

```
USE Master INDEX Master
Repchoice = 0
DO WHILE Repchoice # 5
CLEAR
    @ 2,1  SAY "Master Inventory Report Options"
    @ 2,60 SAY DTOC(T_Date) + "    " + TIME()
    @ 3,0  SAY Uline
    ?
    ?
    TEXT
            1. Entire inventory
            2. Reorder report
            3. On Order report
            4. Purchase orders
            5. Return to master menu
    ENDTEXT
    @ 24,1 SAY "Enter choice(1-5)" ;
      GET Repchoice PICT "9" RANGE 1,5
    READ
```

والاختيارات الثلاثة الأولى تستخدم التقارير التى سبق انشاؤها فى عرض البيانات المطلوبة . أما الاختيار رقم (4) فانه يؤدى الى تشغيل البرنامج (Orders.prg) . وهذا البرنامج يعرض على المستخدم بيانات الأصناف التى تقل عن حد الطلب (Reorder) ، ويتيح له تحديد الكمية المطلوبة من كل صنف . ثم يقوم البرنامج بطباعة أوامر الشراء آليا .

والجزء الثانى يبدأ بانشاء متغيرات الذاكرة (YN) ، (Printer) واعطائهما القيمة (" ") . ثم سؤال المستخدم اذا كان يريد طباعة التقرير على الطابعة أو عرضه على الشاشة ، وذلك بالنسبة للاختيارات التى تقل عن (4) . وذلك لأن الاختيار (4) يتفرع الى برنامج (Orders.prg) الذى يقوم بطباعة أوامر الشراء (Purchase orders) .

وعندما يختار المستخدم الاختيار (Y) فان الشرط الموجود بعد (IF) يتحقق . ويؤدى هذا الى تخزين العبارة (TO PRINT) فى المتغير (Printer) . ويتكون هذا الجزء من السطور التالية :

```
CLEAR
STORE " " TO YN, Printer
IF Repchoice < 4
    @ 5,5  SAY "Send report to printer ?" ;
      GET YN PICT "!"
    READ
    CLEAR

    * - - - - -  Set up for printer.
    IF YN = "Y"
        Printer = "TO PRINT"
    ENDIF
ENDIF(Repchoice < 4)
```

والجزء الثالث يتم فيه اتخاذ القرار تبعا لاختيار المستخدم . فعندما يختار الرقم (1) يتم تكوين التقرير (Allmast) الذى سبق انشاؤه . ويلاحظ استخدام دالة الماكرو (&) للتعويض عن المتغير (Printer) بالقيمة المخزنة فيه ، وهى (TO PRINT) . أى أن الأمر يصبح كالآتى :

```
REPORT FORM Almast TO PRINT
```

وهذا يؤدى الى طباعة التقرير متضمنا بيانات الأصناف الموجودة بالمخزن .

وعندما يختار المستخدم الرقم (2) يتم تكوين التقرير (Reorders) الذى سبق انشاؤه . ويلاحظ أن الأمر المستخدم فى هذه الحالة يستخدم شرطا لتحديد السجلات التى تظهر فى التقرير . وهى سجلات الأصناف التى تقل كميتها عن حد الطلب .

وعندما يختار المستخدم الرقم (3) يتم طباعة التقرير (Onorders) ، وذلك بالنسبة للأصناف التى توجد منها كميات تحت الطلب فقط . وذلك عن طريق استخدام الشرط (On_order >0) .

واستخدام دالة الماكرو (&) فى الاختيارات الثلاث السابقة يفيد فى التحكم فى طباعة التقرير أو عرضه على الشاشة حسب اختيار المستخدم . فاذا أراد المستخدم طباعة التقرير فانه يدخل الحرف (Y) فى المتغير (YN) . وهذا يؤدى الى ادخال العبارة (TO PRINT) فى المتغير

(printer) كما سبق الايضاح . وهذا بالتالي يؤدي الى ادخال عبارة (TO PRINT) بعد الأمر (REPORT FORM) ، مما يؤدي الى طباعة التقرير .

أما اذا أراد المستخدم عرض التقرير على الشاشة فقط دون طباعته ، فانه يدخل أي قيمة أخرى غير (Y) في المتغير (YN) . وبالتالي لايتحقق الشرط بعد (IF) ، ويظل المتغير (printer) خاليا . وهذا يؤدي الى عدم اضافة أي عبارة بعد الأمر (REPORT FORM) . وبالتالي يتم عرضه على الشاشة فقط .

وعندما يختار المستخدم الاختيار (4) ، فان البرنامج يتفرع الى البرنامج الفرعي (Orders) الذي سيتم دراسته فيما بعد .

والجزء الثالث الذي سبق شرحه يتكون من السطور التالية :

```
DO CASE
    CASE Repchoice = 1
        REPORT FORM Allmast & Printer
    CASE Repchoice = 2
        REPORT FORM Reorders FOR (QTY + On_order) ;
          < = Reorder & Printer
    CASE Repchoice = 3
        REPORT FORM Onorder FOR On_order > 0 ;
           & Printer
    CASE Repchoice = 4
        DO Orders
ENDCASE
```

والجزء الرابع والأخير من البرنامج يتم من خلاله ايقاف الشاشة مؤقتا في حالة اختيار المستخدم عرض التقرير على الشاشة وليس على الطابعة . وذلك حتى يستطيع المستخدم قراءة بيانات التقرير على الشاشة . ثم يقوم بالضغط على أي مفتاح للرجوع الى قائمة التقارير مرة أخرى حتى يختار نوعا آخر من التقارير حسب الحاجة أو يختار الرقم (5) للخروج من برنامج التقارير والعودة الى القائمة الرئيسية لبرنامج تشغيل الملف الرئيسي (MMenu.prg) .

١٤ - ٤ - ٧ برنامج أوامر الشراء

لعرض أو طباعة أوامر الشراء (Purchase orders) فان برنامج التقارير (MReports.prg) يتفرع الى برنامج أوامر الشراء

(orders.prg) . وهذا البرنامج يؤدى عدة وظائف . فهو فى البداية يجب أن يبحث خلال الملف الرئيسى (Master.dbf) عن الأصناف التى يجب طلبها . ولتنفيذ ذلك فانه يجمع الكمية الموجودة فعلا (On hand) على الكمية تحت الطلب (On order) ويقارن المجموع بحد الطلب (Reorder) . وفى كل مرة يجد فيها صنفا يقل عن حد الطلب ، فانه يسأل المستخدم عن الكمية المطلوب شراؤها من هذا الصنف . وبعد الانتهاء من ادخال كل الأصناف المطلوب شراؤها ، فانه يطبع أوامر الشراء . وفى نفس الوقت يقوم بتحديث الكمية الموجودة فى الحقل (On_order) فى الملف الرئيسى ، وذلك بجمع الكمية الجديدة على هذا الحقل بالنسبة لكل صنف يتم ادخاله فى أوامر الشراء .

ولتصميم هذا البرنامج نقوم أولا بكتابة الخطوات الأولية (PSEUDOCODE) وهى تكون كالآتى :

١ - يتم مسح الشاشة .
٢ - يتم فتح ملف قاعدة البيانات وملف الفهرس الخاص به .
٣ - يتم تكوين حلقة تكرارية خلال الملف الرئيسى (Master.dbf) .
٤ - يتم اختبار مجموع الكمية الفعلية (on hand) مع الكمية تحت الطلب (On order) ومقارنة المجموع بحد الطلب . وذلك بالنسبة لكل صنف من أصناف الملف الرئيسى (Master.dbf) .
٥ - يتم مسح الشاشة .
٦ - يتم عرض بيانات حالة الصنف الذى يجب طلبه .
٧ - يتم سؤال المستخدم عن الكمية المطلوب طلبها من هذا الصنف .
٨- يتم تحديث الحقل (New_order) الذى يمثل الكمية التى يجب طلبها من الصنف وكذلك تحديث الحقل (Order_date) الذى يمثل تاريخ آخر طلب لهذا الصنف .
٩ - عند ادخال كل الأصناف التى يجب طلبها يتم مسح الشاشة .
١٠- يتم انشاء ملف مؤقت (Temporary File) للأصناف المطلوبة .
١١- يتم تحديث حقل تحت الطلب (On_order) فى الملف الرئيسى (Master.dbf) .
١٢- يتم اعادة محتويات حقل (New_order) الى الصفر .
١٣- يتم فتح الملف المؤقت ، الذى يحتوى على بيانات الأصناف المطلوبة فقط .
١٤- يتم فهرسة هذا الملف بناء على حقل البائع (Vendor) وذلك لتقسيم الأصناف بالنسبة للبائعين .
١٥- يتم تشغيل الطابعة .
١٦- يتم طباعة اسم البائع وعنوانه .
١٧- يتم طباعة الكمية المطلوبة واسم الصنف وسعره لكل صنف من الأصناف

التي يتم شراؤها من هذا البائع .

١٨- عند الانتهاء من هذا البائع ، يتم طباعة السعر الكلي لهذا الطلب كما يتم طباعة اسم المكان المطلوبة له هذه الأصناف وعنوانه .

١٩- يتم الانتقال الى صفحة جديدة على الطابعة ، وكذلك الانتقال الى بائع جديد (Vendor) فى الملف المؤقت .

٢٠- تستمر هذه العملية بالنسبة لجميع البائعين فى الملف المؤقت .

٢١- عند الانتهاء يتم اغلاق الطابعة والعودة الى برنامج التقارير .

١٤ - ٤ - ٨ كتابة البرنامج

بعد كتابة الخطوات الأولية (PSEUDOCODE) للبرنامج يتم كتابة أوامر البرنامج وهى تتلخص فى السطور التالية :

```
* * * * * * * * * * * * * * *  Orders.prg
*     Create Purchase orders for reordering
*     Called from Reports menu, MReports.prg
CLEAR
USE Master INDEX Master
* - -Make the loop,and display goods below reorder
* - -point,and ask the user how many of each to order.
GO TOP
DO WHILE .NOT. EOF()

* - Find out if on_hand plus on_order quantity is
* - - less than reorder point .
IF (Qty + On_order) < = Reorder
        CLEAR

     * - - - - - Show status of item to be reordered
     @ 5,5  SAY "Part number: " + Part_no +" "+ P_name
     @ 6,5  SAY "On hand " + STR(Qty,4)
     @ 7,5  SAY "On order " + STR(On_order,4)
     @ 8,5  SAY "Reorder   " + STR(Reorder,3)
     @ 9,5  SAY "Unit Cost" + STR(Cost,9,2)

     *  Ask user how many to order
     @ 12,5 SAY "Order how many?" GET New_order ;
```

```
        PICT "999"
     REPLACE Order_date WITH T_Date
     READ
ENDIF
SKIP
ENDDO(Continue loop until  end of file)

* - - - When all orders have been placed
* - - - Make a temporary file of items to be ordered.
CLEAR
? "Preparing Files ..... please wait"
?
? "(Prepare printer while waiting)"
COPY TO Temp FOR New_order > 0

* - -Update On_order field in Master file with new
* - -orders,then set the New_order field back to zero.
REPLACE ALL On_order WITH On_order + New_order
REPLACE ALL New_order WITH 0

* - - - -  Use temp file (which contains new orders)
* - - - -  Indexed by vendor
USE Temp
INDEX ON UPPER(Vendor) TO Temp

* - - - - -  Files ready , inform user
CLEAR
? CHR(7)
WAIT "Ready printer and press any key to print orders"
SET PRINT ON
GO TOP

* - - - - -  Loop through Temp file
DO WHILE .NOT. EOF()

    * For each vendor , print name and address
    This_loop = Vendor
    Mtotal = 0
```

```
    ? vendor
    ? Vendor_add
    ?
    ?
    ? "Please send us the following items...."

    * - -For each item to be ordered from this vendor
    * - -print quantity,item , and price .
    DO WHILE Vendor = this_loop .AND. .NOT. EOF()
        ? New_order , P_name , Cost , New_order * Cost
        Mtotal = New_order * Cost + Mtotal
        SKIP
    ENDDO

    * - - - When done with this vendor, print total
    * - - - Cost,and shipping name and address .
    ?
    ? "Total cost :        ", Mtotal
    ?
    ? "Mail to : My company"
    ? "       12 - Tayaran street"
    EJECT
ENDDO(Continue for each vendor in Temp file)
* - - - - when done, turn off printer , and return to
* - - - - Reports menu .
SET PRINT OFF
RETURN
```

والجزء الأول من البرنامج يبدأ بالتعريف باسم البرنامج ووظيفته واسم البرنامج الذى قام باستدعائه . ثم يقوم بمسح الشاشة وفتح الملف الرئيسى (Master.dbf) وملف الفهرس الخاص به . وذلك من خلال السطور التالية :

```
CLEAR
USE Master INDEX Master
```

والجزء الثاني من البرنامج يتم من خلاله انشاء حلقة تكرارية (Loop) للبحث خلال الملف الرئيسي (Master.dbf) عن الأصناف التى يقل مجموع الكمية الموجودة منها والكمية تحت الطلب عن حد الطلب لهذه الأصناف . كما يتم عرض بيانات هذه الأصناف . ثم يتم سؤال المستخدم عن الكمية المطلوب صرفها من كل صنف من هذه الأصناف . ويتم ذلك من خلال السطور التالية :

```
GO TOP
DO WHILE .NOT. EOF()
    * - - - Find out if on_hand plus on_order
    * - - - quantity is less than reorder point
    IF (Qty + On_order) < = Reorder
        CLEAR

        * - - -  Show status of item to be reordered
        @ 5,5 SAY "Part number:" + Part_no + " "+ ;
          P_name
        @ 6,5  SAY "On hand " + STR(Qty,4)
        @ 7,5  SAY "On order " + STR(On_order,4)
        @ 8,5  SAY "Reorder   " + STR(Reorder,3)
        @ 9,5  SAY "Unit Cost" + STR(Cost,9,2)

        *  Ask user how many to order
        @ 12,5 SAY "Order how many?" ;
          GET New_order PICT "999"
        REPLACE Order_date WITH T_Date
        READ
    ENDIF
    SKIP
ENDDO(Continue loop until  end of file)
```

ويلاحظ من هذه الأوامر أن الكمية الجديدة التي يتم طلبها تخزن فى الحقل (New_order) . كما أن حقل تاريخ الطلب (Order_date) يتم تغييره بتاريخ اليوم الحالى الذى يتم فيه طلب الأصناف . وهو التاريخ الذى يتم ادخاله عند تشغيل نظام المخازن من البداية .

والجزء الثالث من البرنامج يتم من خلاله نسخ جميع بيانات الأصناف المطلوبة فى ملف مؤقت (temporary file) يسمى (temp) .

ويتم ذلك من خلال السطور التالية :

```
CLEAR?
? "Preparing Files ..... please wait"
?
? "(Prepare printer while waiting)"
COPY TO Temp FOR New_order > 0
```

والجـــزء الرابع يتم من خلاله تعديل بيانات الملف الرئيسى (Master.dbf) بناء على الكميات الجديدة التى تم طلبها من بعض الأصناف . حيث يتم تعديل بيانات حقل الكمية تحت الطلب (On_order) باضافة الكمية الجديدة التى تم طلبها الى الكمية السابقة . ثم يتم اعادة حقل الكمية الجديدة (New_order) الى الصفر. ويتم ذلك من خلال السطور التالية :

```
* - -Update On_order field in Master file with new
* - -orders,then set the New_order field back to zero.
REPLACE ALL On_order WITH On_order + New_order
REPLACE ALL New_order WITH 0
```

والجزء الخامس يتم من خلاله فتح الملف المؤقت (Temp) . كما يتم فهرسته على حقل البائع (Vendor) . وذلك حتى يمكن طباعة طلب الشراء الخاص بكل بائع على حدة . ويتم ذلك من خلال السطور التالية :

```
USE Temp
INDEX ON UPPER(Vendor) TO Temp
```

والجزء السادس يتم من خلاله توجيه المستخدم لتجهيز الطابعة ثم تشغيل الطابعة وانشاء حلقة تكرارية لطباعة أوامر الشراء لكل بائع (Vendor) على حدة . كما يتم انشاء حلقة تكرارية أخرى داخلها لطباعة الأصناف الخاصة بكل بائع فى طلب الشراء الخاص به . ويتم ذلك من خلال السطور التالية :

```
CLEAR?
CHR(7)
WAIT "Ready printer and press any key to print orders"
SET PRINT ON
GO TOP
```

```
* - - - - - Loop through Temp file
DO WHILE .NOT. EOF()
```

ومن خلال الحلقة التكرارية الأولى ، يتم تخزين اسم البائع فى متغير ذاكرة (This_loop) . كما يتم انشاء متغير ذاكرة (Mtotal) لحساب المجموع الكلى لأسعار الأصناف فى كل طلب شراء . كما يتم طباعة اسم البائع وعنوانه فى بداية طلب الشراء . والسطور التالية توضح ذلك :

```
This_loop = Vendor
Mtotal = 0
? vendor
? Vendor_add
?
?
? "Please send us the following items...."
```

ومن خلال الحلقة التكرارية الثانية يتم عرض بيانات الأصناف الخاصة بكل بائع (Vendor) على حدة ، وذلك بالنسبة لكل بائع يتم تخزين اسمه فى المتغير (This_loop) . والسطور التالية توضح هذه الحلقة :

```
DO WHILE Vendor = this_loop .AND. .NOT. EOF()
    ? New_order ,P_name , Cost , New_order * Cost
    Mtotal = New_order * Cost + Mtotal
    SKIP
ENDDO
```

وعند الانتهاء من بائع معين ، أى قبل الانتقال الى اسم بائع جديد ، يتم طباعة أمر الشراء . كما يتم طباعة التكلفة الكلية لهذا الطلب والعنوان الذى يتم ارسال الأصناف اليه . ثم يتم نقل ورقة الطباعة (Eject) .

والسطور التالية توضح هذه العملية :

```
?  "Total cost :          ", Mtotal
?
? "Mail to : My company"
? "         12 - Tayaran street"
EJECT
```

ثم تستمر الحلقة التكرارية الخارجية حتى يتم طباعة باقى أوامر الشراء . وبعد ذلك يتم اعادة الطابعة الى وضعها الأصلى (Off) والعودة الى البرنامج الرئيسى (Master.dbf) ، وذلك من خلال السطور التالية :

```
ENDDO(Continue for each vendor in Temp file)
SET PRINT OFF
RETURN
```

١٤ - ٥ برنامج تعديل الملف الرئيسى

يتم الانتقال الى برنامج تعديل الملف الرئيسى عن طريق اختيار الرقم (3) فى قائمة برنامج تشغيل الملف الرئيسى . حيث يتم تنفيذ البرنامج (MEdit.prg). والخطوات الأولية (PSEUDOCODE) لهذا البرنامج تكون كالآتى :

١- يتم فتح الملف الرئيسى (Master.dbf) وملف الفهرس الخاص به .
٢- يتم تكوين حلقة تكرارية لاجراء التعديل .
٣- البحث عن رقم الجزء (Part number) المطلوب تعديل بياناته .
٤ - عند عدم ادخال رقم الجزء (Part number) ، يتم الرجوع الى القائمة .
٥ - عند العثور على الجزء المطلوب ، يتم عرض بياناته باستخدام شاشة الادخال (IScreen1) .
٦ - اذا لم يتم العثور على الجزء المطلوب ، يتم تحذير المستخدم واعطاؤه الفرصة لادخال رقم جديد .
٧- الاستمرار فى التعديل حتى يطلب المستخدم الخروج .
٨- عند الانتهاء يتم الرجوع الى القائمة الرئيسية لتشغيل الملف الرئيسى .

والبرنامج الذى يحقق هذه الخطوات الأولية يتم كتابته كالآتى :

```
******************************************* MEdit.prg
*    Edit the inventory Master file
*    Called from Master menu , MMenu.prg .
USE Master INDEX Master

* - - - - -  Start loop for performing edits
Partnumb = "X"
DO WHILE Partnumb # " "

    * - - - Find out what part number to edit.
```

```
CLEAR
@ 2,1  SAY "Edit Inventory Master File"
@ 2,60 SAY DTOCT(T_Date) + "   " + TIME()
@ 3,0  SAY Uline
?
?
?
?
Partnumb = SPACE(5)
@ 15,5 SAY "Edit for what part number?" ;
   GET Partnumb
READ

* - - - - - -  Try to find that part number
Partnumb = UPPER(Partnumb)
SEEK Partnumb
DO CASE

    * - - - If no part number entered, return to
    * - - - Master menu.
    CASE Partnumb = " "
        CLEAR

    * - - - If part number found , edit using
    * - - - Iscreen1 format file
    CASE FOUND()
        SET FORMAT TO Iscreen1
        READ
        SET FORMAT TO

    * - - - Otherwise warn user, and allow another
    * - - - try
    CASE .NOT. FOUND()
        @ 17,5 SAY "There is no part" + "Partnumb"
        @ 24,5 SAY "Press any key to try again"
        WAIT
 ENDCASE
ENDDO(Continue editing untill user requests ...exit)
```

```
* - - - - -  Return to Master menu
RETURN
```

والجزء الأول من البرنامج يبدأ كالعادة بالتعريف باسم البرنامج ووظيفته ثم اسم الملف الذى يقوم باستدعائه . ثم يتم فتح الملف الرئيسي وملف الفهرس الخاص به . وكالعادة يتم تكوين حلقة تكرارية للبحث عن رقم الجزء (Part number) الذى يدخله المستخدم فى المتغير (Partnumb) ويتم ذلك من خلال السطور التالية :

```
USE Master INDEX Master
Partnumb = "X"
DO WHILE Partnumb # " "
```

ثم يتم عرض عنوان الشاشة والتاريخ والوقت ثم سؤال المستخدم عن الرقم المطلوب . وذلك من خلال السطور التالية :

```
CLEAR
@ 2,1  SAY "Edit Inventory Master File"
@ 2,60 SAY DTOCT(T_Date) + "    " + TIME()
@ 3,0  SAY Uline
?
?
Partnumb = SPACE(5)
@ 15,5 SAY "Edit for what part number?" GET Partnumb
READ
```

ثم يتم تحويل رقم الجزء الى حروف كبيرة (Uppercase) والبحث عنه بواسطة الأمر (SEEK) . وذلك كالآتى :

```
Partnumb = UPPER(Partnumb)
SEEK Partnumb
```

ثم يبدأ البرنامج بعد ذلك فى اتخاذ القرار بناء على رقم الجزء الذى يدخله المستخدم . فاذا لم يدخل المستخدم أى رقم يتم مسح الشاشة والخروج من الحلقة التكرارية مما يؤدى الى الرجوع الى القائمة الرئيسية لتشغيل الملف الرئيسي . أما اذا أدخل رقما وتم العثور على هذا الرقم (FOUND()) فان البرنامج يعرض بيانات هذا الجزء من خلال شاشة الادخال (IScreen1) التى سبق انشاؤها . ثم يتيح للمستخدم تعديل بيانات هذا الجزء . ثم يتم اغلاق ملف شاشة الادخال

.(IScreen1)

واذا أدخل المستخدم رقما ، ولكن لم يتم العثور على هذا الرقم فان البرنامج يحذر المستخدم ويعطيه الفرصة لادخال رقم آخر .

والسطور التالية توضح هذه العملية :

```
DO CASE
    CASE Partnumb = " "
        CLEAR
    CASE FOUND()
        SET FORMAT TO Iscreen1
        READ
        SET FORMAT TO
    CASE .NOT. FORUND()
        @ 17,5 SAY "There is no part " + Partnumb
        @ 24,5 SAY "Press any key to try again"
        WAIT
ENDCASE
ENDDO(Continue editing until  user requests exit)
RETURN
```



الفصل الخامس عشر

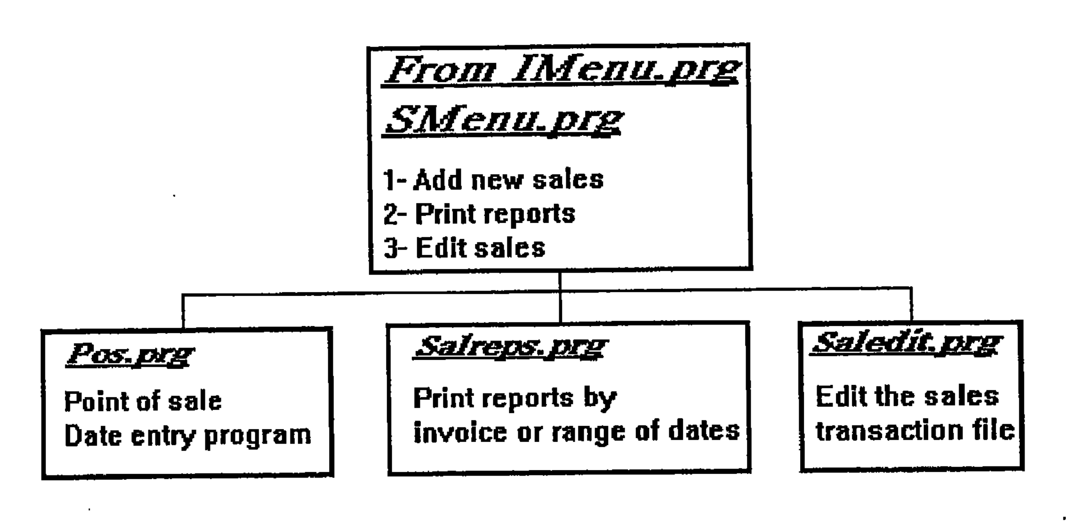
برنامج تشغيل ملف المبيعات



هذا الجزء من برنامج المخازن يختص بمتابعة مبيعات الأصناف وتحديث ملف المبيعات بناء على ذلك . ويتم ذلك عن طريق برنامج مكون من أربعة برامج منفصلة كما سيتم الايضاح فيما بعد. ويقوم هذا البرنامج بالاشراف على نقطة البيع (Point of Sale) ، حيث يقوم بتجهيز فواتير البيع بناء على رقم الجزء الذى يدخله المستخدم . ويقوم البرنامج كذلك بملء بيانات الأصناف فى الفاتورة آليا . كما يقوم البرنامج بتحذير المستخدم اذا أدخل رقم جزء غير موجود . ويتيح له ادخال رقم جديد . كما يقوم البرنامج بعد ذلك بتجميع أسعار الأجزاء فى الفاتورة بالاضافة الى تخزين أرقام الفواتير .

١٥ - ١ تركيب برنامج المبيعات

يتكون هذا البرنامج من أربعة برامج كما سبق الايضاح . وهى برنامج (SMenu.prg) الذى يتحكم فى البرامج الثلاث الأخرى ، وبرنامج (Pos.prg) الذى يتحكم فى نقطة البيع (Point of sale) ، وبرنامج (Salreps.prg) الذى يطبع التقارير التى توضح موقف مبيعات الأصناف ، وبرنامج (Saledit.prg) الذى يسمح بتعديل بيانات ملف المبيعات . والتركيب الهرمى للبرنامج يتضح من الشكل التالى :



شكل (١٥ - ١)

وفى الجزء التالى سوف يتم شرح برامج (SMenu) ، (Pos) ، (Salreps) مع تأجيل شرح برنامج (Saledit) الى الفصل الخاص بتحديث الملفات .

١٥ - ٢ برنامج القائمة الرئيسية (Smenu.prg)

يتم تشغيل هذا البرنامج عندما يختار المستخدم الاختيار رقم (2) فى القائمة الرئيسية لنظام المخازن . وعند تشغيل البرنامج تظهر القائمة التالية على الشاشة .

```
Sales System Menu     02/20/90     04:50:30

  1 .   Enter point of sale routine
  2 .   Print sales reports
  3 .   Edit sales data
  4 .   Return to main menu

      Enter choice (1 - 4 )
```

شكل (١٥ - ٢)

والبرنامج لايختلف تركيبه عن أى برنامج قائمة من البرامج السابقة . ولذلك فليست هناك حاجة الى كتابة الخطوات الأولية (PSEUDOCODE)له ، ونكتفى فقط بعرض سطور البرنامج كالآتى :

```
* * * * * * * * * * * * * * *   * * * * * * * SMenu.prg
*    Menu of sales portion of the inventroy system
*    Called from inventory system Main menu .

* - - - - -  Set up loop for presenting menu
Schoice = 0
DO WHILE Schoice # 4
    CLEAR
    @ 2,1  SAY "Sales system menu"
    @ 2,60 SAY DTOC(T_Date) + "  " + TIME()
    3,0  SAY Uline
```

```
    ?
    ?
    TEXT

        1. Enter point of sale routine
        2. Print sales reports
        3. Edit sales data
        4. Return to main menu

    ENDTEXT
    @ 24,1 SAY "Enter choice (1 - 4) " ;
       GET Schoice PICT "9"
    READ

    * - - - - -  Branch to appropriate choices
    DO CASE
        CASE Schoice = 1
            DO Pos
        CASE Schoice = 2
            DO Salreps
        CASE Schoice = 3
            DO Saledit
    ENDCASE
ENDDO(While schoice #4)
* - - - - -  when done , return to main menu
RETURN
```

١٥ - ٢ برنامج نقطة البيع (Pos.prg)

قبل البدء في شرح أوامر هذا البرنامج سيتم أولا شرح ما يظهر على الشاشة عند تنفيذه حتى يكون القارىء متتبعا لوظائف هذا البرنامج . فعند تشغيل البرنامج تظهر شاشة تشبه فاتورة البيع تماما كما هو واضح من الشكل (١٥ - ٢) .

02/20/90	Invoice number :
Clerk	Customer :

Part #	Name	Qty	Price	Total

شكل (١٥ - ٣)

والبرنامج يملأ البيانات الموجودة أعلى الفاتورة آليا مثل التاريخ ورقم الفاتورة كما يقوم المستخدم بكتابة اسم الموظف القائم بعملية البيع (Clerk) واسم العميل الذى يتم البيع له . ثم يظهر عمود ضوئى لادخال رقم الجزء فيه كما يتضح من الشكل (١٥ - ٤) .

02/20/90	Invoice number :
Clerk	Customer :

Part #	Name	Qty	Price	Total

شكل (١٥ - ٤)

وعند ادخال المستخدم لرقم جزء غير موجود يظهر التحذير التالى :
No such part !!
ويمكن للمستخدم فى هذه الحالة أن يحاول مرة ثانية . وعند كتابة رقم جزء موجود فان اسم هذا الجزء يظهر آليا ، كما تظهر أعمدة ضوئية (Highlights) لادخال الكمية (Qty)، وسعر البيع(Selling Price).كما يقوم البرنامج بحساب السعر الكلى عن طريق ضرب الكمية (Qty) فى سعر البيع (Price) ثم يظهر العمود الضوئى الخاص برقم الجزء التالى حتى يقوم المستخدم بادخال رقم جديد . وتتضح هذه العملية من الشكلين التاليين .

02/20/90		Invoice number :		
Clerk		Customer :		
Part #	Name	Qty	Price	Total
A-121	Shoe	30	20	

شكل (١٥ - ٥)

02/20/90		Invoice number :		
Clerk		Customer :		
Part #	Name	Qty	Price	Total
A-121	Shoe	30	20	600

شكل (١٥ - ٦)

وعند ادخال المستخدم لرقم جزء جديد موجود فى المخزن بالاضافة الى ادخال الكمية والسعر لهذا الصنف ، يقوم المستخدم بتنفيذ نفس العملية السابقة ويتضح ذلك من الشكل التالى :

02/20/90 Invoice number :

Clerk Customer :

Part #	Name	Qty	Price	Total
A-121	Shoe	30	20	600
A-122	Shirt	20	25	500

شكل (١٥ - ٧)

وعندما يدخل المستخدم رقما موجودا ولكنه ليس الجزء الذى يريده (وهذا يتضح له من اسم الجزء الذى يظهر آليا) فيمكنه فى هذه الحالة الضغط على مفتاح (<--) مرتين للرجوع الى مكان رقم الجزء وكتابة رقم جزء آخر . كما يمكن الضغط على مفتاح الادخال مرتين لتنفيذ نفس العملية السابقة ، حيث أن الضغط على مفتاح الادخال دون ادخال أى عدد فى الكمية (Qty) أو فى السعر (Price) يخبر البرنامج أن هذا الجزء غير مطلوب ، وبالتالى يعود المؤشر الى مكان رقم الجزء لادخال رقم جزء جديد .

وعندما ينتهى المستخدم من ادخال بيانات هذه الفاتورة (Invoice) ، فانه يضغط على مفتاح الادخال عندما يكون المؤشر واقفا على العمود الضوئى الخاص برقم الجزء (Part #) . وفى هذه الحالة يقوم البرنامج بتجميع السعر الكلى للفاتورة كما يعرض رسالة للمستخدم لسؤاله اذا كان يريد طباعة هذه الفاتورة أم لا . ويتضح هذا من الشكل التالى :

02/20/90 Invoice number :

Clerk ▒▒▒▒ Customer : ▒▒▒▒

Part #	Name	Qty	Price	Total
A-121	Shoe	30	20	600
A-122	Shirt	20	25	500

Total : 1100

Print invoice ? (Y/N)

شكل (١٥ - ٨)

وعندما يكتب المستخدم (Y) يتم طباعة هذه الفاتورة (Invoice) ، ثم يقوم البرنامج بمسح الشاشة وعرض سؤال للمستخدم اذا كان يريد طباعة فاتورة أخرى فاذا أجاب بنعم (Y) يتم عرض فاتورة جديدة بنفس الطريقة وبرقم فاتورة جديد . واذا أجاب بلا (N) يتم الرجوع الى قائمة البيع .

١٥ - ٣ - ١ كتابة الخطوات الأوليه (PSEUDOCODE)

كما سبق الايضاح فان البرنامج الخاص بالتحكم فى نقطة البيع (Point of sale) يتم تسميته (Pos.prg) . ويتم كتابة الخطوات الأولية له كالآتى :

١ - يتم الحصول على آخر رقم فاتورة من ملف المبيعات (Sales.dbf) .
٢- يتم انشاء ملف مؤقت (Temporary File) لمتابعة حركة البيع .
٣- يتم فتح الملف الرئيسى والملف المؤقت .
٤- يتم ربط الملف الرئيسى بالملف المؤقت .

٥- يتم تكوين حلقة تكرارية لعرض الفواتير على الشاشة . حيث يتم أولا عرض السطور الأولى من التقرير .
٦- يتم تكوين حلقة تكرارية لكل جزء يتم ادخاله الى الفاتورة .
٧- يتم تكوين حلقة تكرارية لاختبار رقم الجزء والتأكد من وجوده فى الملف الرئيسي (Master.dbf) .
٨- عند عدم ادخال أى رقم يتم الخروج من البرنامج .
٩- عند ادخال رقم غير موجود يتم تحذير المستخدم لادخال رقم جديد .
١٠- يتم ادخال الكمية والسعر لكل جزء .
١١- عند ادخال كمية (صفر) يتم ادخال رقم جديد .
١٢- يتم حساب السعر الكلي للصنف وعرضه تحت العنوان (Total) .
١٣- يتم تخزين هذه الحركة في ملف مؤقت (Temporary File) .
١٤- يتم زيادة عداد رقم السطر لعرض بيانات الصنف التالى .
١٥- عند الوصول الى آخر سطر في الشاشة يتم زحزحة الشاشة (Scrolling) سطرا واحدا الى أعلى .
١٦- يتم الاستمرار فى تنفيذ الحلقة التكرارية حتى يتوقف المستخدم عن ادخال الأصناف . وفى هذه الحالة يتم عرض السعر الكلى للفاتورة (Grand Total) ثم يتم عرض سؤال للمستخدم عما اذا كان يريد طباعة الفاتورة أم لا .
١٧- يتم سؤال المستخدم اذا كان يريد طباعة فاتورة أخرى ، وبناء على ذلك تستمر الحلقة التكرارية حتى يطلب المستخدم الخروج .
١٨- يتم اغلاق ملفات قواعد البيانات (Database Files) كما يتم تحديث ملف المبيعات (Sales File).
١٩- يتم مسح الملف المؤقت (Temporary File) .
٢٠- يتم الرجوع الى القائمة الرئيسية للمبيعات .

١٥ - ٣ - ٢ كتابة البرنامج

قبل كتابة البرنامج يجب ملاحظة أن هذا البرنامج طويل بعض الشيء . وربما يحتاج الى برنامج معالج كلمات آخر غير البرنامج الموجود مع برنامج (+ DBase III) وذلك فى حالة كتابته متضمنا كل سطور الملاحظات . ولذلك يفضل عدم كتابة سطور الملاحظات خلال البرنامج اذا تم استخدام المصحح الخطى الخاص ببرنامج (+DBase III) . ولكن سنضيف سطور الملاحظات هنا للتوضيح فقط .

ملاحظة

عند الانتهاء من جميع البرامج الخاصة بنظام المخازن يمكن مسح جميع سطور الملاحظات وذلك بعد نسخ البرامج الأصلية فى قرص آخر . حيث أن

ذلك يؤدى الى سرعة تنفيذ البرنامج بدرجة كبيرة .

ويتم كتابة سطور البرنامج كالآتى :

```
* * * * * * * * * * * * * * * Pos.prg
*    Point of sale data entry program for sales
*    Called from Sales menu , SMenu.prg

* - Get last-used invoice number from the Sales file.
USE Sales
GO BOTT
Minvoice = Invoice_no

* - - - - - -  Add new transactions to a temporary file
SET SAFETY OFF
COPY STRUCTURE TO Tempinv

* - - - - - -  Open Master and temporary files
SELECT A
USE Master INDEX Master
SELECT B
USE Tempinv
SET RELATION TO part_no INTO Master
* - - - - - -  Set pointer in temporary file
StartTrans = 1

* - - - - - -  Set up loop for displaying invoice forms
Again = "Y"
DO WHILE Again = "Y"
    * - - Set up top portion of invoice on the screen
    CLEAR
    Minvoice = Minvoice + 1
    STORE SPACE (30) TO MClerk, MCust
    Mtotal = 0
    @ 1,2  SAY T_Date
    @ 1,30 SAY "Invoice number:" + STR(Minvoice,5)
    @ 2,2  SAY "Clerk" GET MClerk
```

```
@ 2,35 SAY "Customer:" GET MCust
@ 3,0  SAY Uline
? "PART #    Name", SPACE(20)
?? "Qty      Price      Total"
READ
* - - - Set up loop for each item in the invoice
Row = 7
Adding = .T.
DO WHILE Adding
    Partnumb = SPACE(5)
    ok = .F.
    * - - - loop to check validity of part number
    DO WHILE .NOT. ok
        * - - - Set up invoice memory variables.
        Quantity = 0.00
        Sel_Price = 0.00
        * - - - - - -  Ask for part number.
        Partnumb = SPACE(5)
        @ Row,2 GET Partnumb
        READ
        * - - - - - -  Make sure Part number exixts.
        partnumb = UPPER(TRIM(Partnumb))
        SELECT A
        SEEK Partnumb

        * - - - - - -  Decide next step based on
        * - - - - - -  existance of part number.
        DO CASE
            * - - - - - -  No part number entered
            CASE LEN(partnumb) = 0
                ok = .T.
                Adding = .F.
            * - - - - - Part number does not exist.
            CASE .NOT. FOUND()
                @ Row, 10 SAY "No such part !!"
                oK = .F.
            * - - - - - -  Part number exists
            CASE FOUND()
```

```
* - - - - -Display Part name, get
* - - - - - -  quantity and price
@ Row,10 SAY P_name
@ Row,35 GET Quantity PICT "999.99"
@ Row,40 GET Sel_Price PICT "999.99"
READ
* - - - If quantity is zero , loop
* - - - Else , Display total.
IF Quantity = 0
   LOOP
ESLE
  @ Row,50 SAY Quantity * Sel_Price;
   PICT "##,###.##"
  Mtotal = Mtotal + Quantity * ;
   Sel_price
   ok = .T.
ENDIF

* - - - Add a blank record to the
* - - - Tempinv file , and fill in
* - - - the fields.
SELECT B
APPEND BLANK
REPLACE Date WITH T_Date
REPLACE Clerk WITH M_Clerk
REPLACE Invoice_no WITH Minvoice
REPLACE Customer WITH MCust
REPLACE Part_no WITH Partnumb
REPLACE Qty WITH Quantity
REPLACE Price WITH Sel_Price
REPLACE Posted WITH .F.
    ENDCASE
ENDDO(continue loop for checking part number)
Row = Row + 1
* - - - Scroll screen if reached end of screen.
IF Row >= 19
    @ 24,1
    ?
```

```
         Row = 19
       ENDIF
    ENDDO(while still adding items to invoice)
    * - - - Display grand total,and pause before next
    * - - - - - -  invoice.
    @ Row+2,40 SAY "Total:"
    @ Row+2,50 SAY Mtotal PICT "##,###.##"
    Pinvoice = "Y"
    @ 23,2 SAY "Print invoice ? (Y/N)" GET Pinvoice ;
        PICT "!"
    READ
    * - - - - - -  Print invoice , Reset Start Trans.
    IF Pinvoice = "Y"
        SET PRINT ON
        ? "Date:" , T_Date
        ? "Invoice number:" , STR(Minvoice,5)
        ? "Customer:" , MCust , SAPACE(20)
        ?? "Clerk:" , Mclerk
        ? Uline
        ?
        SELECT B
        GOTO StartTrans
        LIST OFF WHILE .NOT. EOF() Part_no, ;
        A --> P_name, Qty , Price, Qty * Price
        ?
        ?
        ? "Total:" , SPACE(34), Mtotal
        EJECT
        SET PRINT OFF
        StartTrans = RECCOUNT() + 1
    ENDIF
    CLEAR
    @ 23,2 SAY "Do another transaction? (Y/N)" ;
        GET Again PICT "!"
    READ
ENDDO(add invoices while user does not request exit)
```

```
* - - - - -  Close databases and update sales file.
CLOSE DATABASES
CLEAR
? "Updating transaction file , please wait ..."
SET TALK ON
USE SALES
APPEND FROM Tempinv
USE Tempinv
ZAP
USE Sales INDEX Sales
REINDEX
SET TALK OFF
CLOSE DATABASES
RETURN
```

والجزء الأول من البرنامج يبدأ كالعادة بالتعريف باسم البرنامج ووظيفته ثـــم اسم البرنامج الذى قام باستدعائه . ثم يتم فتح ملف المبيعات (Sales.dbf) دون استخدام أى فهرس معه . وذلك للحصول على آخر رقم فاتورة (Invoice_no) . حيث أن رقم الفاتورة يتوقف على ترتيب ادخال هذه الفاتورة في الملف . ثم يتم تخزين هذا الرقم فى المتغير (Minvoice) . ويتم ذلك من خلال السطور التالية :

```
USE Sales
GO BOTT
Minvoice = Invoice_no
```

والجزء الثاني يوضح استخدام الملف المؤقت (Tempinv.dbf) فى تخزين الفواتير الجديدة قبل نقلها الى ملف المبيعات (Sales.dbf) . وتوفر هذه الطريقة سرعة كبيرة لتنفيذ البرنامج . وبعد الانتهاء من ادخال الفواتير المطلوبة ، يقوم البرنامج باضافـــة هذه الفواتير الى ملف المبيعات . ويلاحظ هنا استخدام الأمر (SET SAFETY OFF) حتى يتم نسخ هيكل ملف المبيعات فى الملف المؤقت دون ظهور رسالة التحذير المعتادة فى هذه الحالة والسطور التالية توضح هذه العملية .

```
SET SAFETY OFF
COPY STRUCTURE TO Tempinv
```

والجزء الثالث يتم من خلاله فتح الملف المؤقت والملف الرئيسي فى مناطق عمل مختلفة . ثم يتم ربط الملفين بناء على حقل رقم الجزء (Part number) . وهذا يساعد على اختبار رقم الجزء الذى يدخله المستخدم والتأكد من وجوده فى الملف الرئيسى . كما أن الربط بين الملفين يساعد بعد ذلك على الحصول على اسم الجزء حتى يتم كتابته فى الفاتورة . والسطور التالية توضح فتح هذين الملفين وانشاء العلاقة (Relation) بينهما .

```
SELECT A
USE Master INDEX Master
SELECT B
USE Tempinv
SET RELATION TO part_no INTO Master
```

وعند طباعة الفواتير يجب أن يعرف البرنامج رقم السجل الذى يبدأ بطباعته . لذلك يتم انشاء متغير الذاكرة (StartTrans)واعطاؤه القيمة (1) من خلال السطر التالى .

```
StartTrans = 1
```

والحزء الرابع يتم من خلاله تكوين حلقة تكرارية لاضافة الفواتير (Invoices) ويتم استخدام المتغير (Again) فى التحكم فى هذه الحلقة كما يتم من خلال الحلقة زيادة قيمة المتغير (Minvoice) بواحد عند الانتقال الى فاتورة جديدة .

كما يتم انشاء المتغير (MClerk) ، والمتغير (MCust) لكل فاتورة جديدة لتسجيل اسم الموظف القائم بالبيع واسم العميل فى كل فاتورة . ثم يتم عرض السطور العلوية للتقرير . وذلك من خلال السطور التالية :

```
* - - - - - Set up loop for displaying invoice forms
Againg = "Y"
DO WHILE Again = "Y"
    * - - Set up top portion of invoice on the screen
    CLEAR
    Minvoice = Minvoice + 1
    STORE SPACE (30) TO Mclerk, MCust
    Mtotal = 0
    @ 1,2  SAY T_Date
```

```
@ 1,30 SAY "Invoice number:" + STR(Minvoice,5)
@ 2,2  SAY "Clerk" GET MClerk
@ 2,35 SAY "Customer:" GET MCust
@ 3,0  SAY Uline
? "PART #    name", SPACE(20)
?? "Qty      Price      Total"
READ
```

والجزء الخامس يتم من خلاله تكوين حلقة تكرارية لادخال الأصناف في الفاتورة . ويتم انشاء المتغير (Adding) للتحكم في هذه الحلقة . ويبدأ عرض الأصناف ابتداء من السطر رقم (7) على الشاشة . وتستمر الحلقة التكرارية حتى يضغط المستخدم على مفتاح الادخال دون ادخال رقم جزء (Part number) . والسطور التالية توضح هذه العملية .

```
Row = 7
Adding = .T.
DO WHILE Adding
```

والجزء السادس يتم من خلاله تكوين حلقة تكرارية أخرى لاختبار رقم الجزء الذى يدخله المستخدم . وذلك بعد انشاء متغير الذاكرة (Partnumb) . وكذلك انشاء المتغير المنطقي (ok) للتحكم في الحلقة التكرارية . ويتم اعطاء هذا المتغير القيمة (.F.) حتى يتم تنفيذ الحلقة التكرارية مرة واحدة على الأقل . والسطور التالية توضح هذه العملية .

```
Partnumb = SPACE(5)
ok = .F.
DO WHILE .NOT. ok
```

كما يتم انشاء متغير ذاكرة (Quantity) لتخزين الكمية المباعة من الصنف، والمتغير (Sel_Price) لتخزين سعر هذا الصنف . وذلك من خلال السطور التالية .

```
Quantity = 0.00
Sel_Price = 0.00
```

واستخدام نقطة العلامة العشرية هنا مهم لتخزين الأعداد متضمنة رقمين عشريين .

والجزء السابع يتم من خلاله سؤال المستخدم عن رقم الجزء المطلوب ادخاله . ويتم تخزين هذا الرقم فى المتغير (Partnumb)، ثم يتم تحويل هذا الرقم الى حروف كبيرة (Uppercase) حتى يتم البحث عنه بواسطة الأمر (SEEK) . والسطور التالية توضح هذه العملية .

```
Partnumb = SPACE(5)
@ Row,2 GET Partnumb
READ
Partnumb = UPPER(TRIM(Partnumb))
SELECT A
SEEK Partnumb
```

والجزء الثامن يتم من خلاله اتخاذ القرار بناء على ما يدخله المستخدم فى المتغير (Partnumb) . فعندما يضغط المستخدم على مفتاح الادخال دون ادخال أى رقم جزء يتم تخزين القيمة (.T.) فى المتغير (ok) والقيمة (.F.) فى المتغير (Adding) وذلك للخروج من الحلقة التكرارية . حيث أن ذلك معناه أن المستخدم يريد الخروج . والسطور التالية توضح هذه العملية .

```
DO CASE
    CASE LEN(partnumb) = 0
        ok = .T.
        Adding = .F.
```

وعندما يتم ادخال رقم جزء غير موجود فى الملف يتم تحذير المستخدم واعطاؤه الفرصة للمحاولة مرة ثانية . ويتم ذلك عن طريق تخزين القيمة (.F.) فى المتغير (ok) حتى يتم تنفيذ الحلقة التكرارية مرة ثانية . ويتم ذلك من خلال السطور التالية :

```
CASE .NOT. FOUND()
@ Row, 10 SAY "No such part !!"
oK = .F.
```

وعندما يتم ادخال رقم موجود فى الملف ، فان البرنامج يأتى بباقى البيانات الخاصة بهذا الصنف مثل اسم الصنف (P_name) ثم يسأل عن

الكمية المطلوبة من هذا الصنف وسعر البيع . ويتم ذلك من خلال السطور التالية :

```
CASE FOUND()
    * - - - - - -  Display Part name, get
    * - - - - - -  quantity and price
    @ Row,10 SAY P_name
    @ Row,35 GET Quantity PICT "999.99"
    @ Row,40 GET Sel_Price PICT "999.99"
    READ
```

والجزء التاسع يتم من خلاله حساب السعر لكل صنف ثم حساب السعر الكلى للفاتورة . كما يتم من خلال هذا الجزء أيضا اعادة تنفيذ الحلقة التكرارية باستخدام الأمر (LOOP) فى حالة ادخال المستخدم للكمية (صفر) . وذلك عندما يضغط على مفتاح الادخال عند وقوف المؤشر على العمود الضوئى (Highlight) الخاص بالكمية . ويتم ذلك من خلال السطور التالية :

```
IF Quantity = 0
    LOOP
ESLE
    @ Row,50 SAY Quantity * Sel_Price;
       PICT "##,###.##"
    Mtotal = Mtotal + Quantity * Sel_Price
    ok = .T.
ENDIF
```

والجزء العاشر يتم من خلاله نقل البيانات التى تم ادخالها فى متغيرات الذاكرة الى الحقول المقابلة فى الملف المؤقت (Tempinv) . وذلك من خلال السطور التالية :

```
SELECT B
APPEND BLANK
REPLACE Date WITH T_Date
REPLACE Clerk WITH M_Clerk
REPLACE Invoice_no WITH Minvoice
REPLACE Customer WITH MCust
REPLACE Part_no WITH Partnumb
REPLACE Qty WITH Quantity
```

```
REPLACE Price WITH Sel_Price
REPLACE Posted WITH .F.
ENDCASE
ENDDO(continue loop for checking part number)
```

والجزء التالى يتم من خلاله زيادة عداد السطور بواحد لكتابة بيانات الصنف التالى . وعند الوصول الى نهاية الشاشة يتم زحزحة الشاشة (Scrolling) بمقدار سطر واحد لأعلى . ويتم ذلك عن طريق تحريك المؤشر الى آخر سطر على الشاشة دون كتابة أى شىء فيه ، وذلك مع تثبيت رقم السطر (Row) عند الرقم (١٩) . ويتم ذلك من خلال السطور التالية :

```
Row = Row + 1
* - - - - -  Scroll screen if reached end of screen.
IF Row >= 19
    @ 24,1
    ?
    Row = 19
ENDIF
ENDDO(while still adding items to invoice)
```

وعند الانتهاء من ادخال الأصناف فى هذه الفاتورة ، يعرض البرنامج السعر الكلى (Total) لهذه الفاتورة ويسأل المستخدم اذا كان يريد طباعة الفاتورة أم لا . والسطور التالية توضح هذه العملية :

```
@ Row+2,40 SAY "Total:"
@ Row+2,50 SAY Mtotal PICT "##,###.##"
Pinvoice = "Y"
@ 23,2 SAY "Print invoice ? (Y/N)" ;
   GET Pinvoice PICT "!"
READ
```

وعندما يكتب المستخدم (Y) يتم تشغيل الطابعة . ثم يتم كتابة رأس التقرير (Heading) . ثم يتم تحريك مؤشر السجلات (Record pointer) الى أول سجل فى الملف المؤقت (Tempinv) . وهو السجل الذى تم تخزين رقمه فى المتغير (StartTrans) . ويتم عرض بيانات الصنف فى الفاتورة وحساب السعر الكلى لهذا الصنف ثم

الانتقال الى الصنف التالى وعرض بياناته وهكذا حتى نهاية الملف . وعند الانتهاء من ادخال كل الأصناف فى الفاتورة يتم تقديم صفحة جديدة على الطابعة باستخدام الأمر (EJECT) للتجهيز لطباعة فاتورة جديدة عندما يريد المستخدم ذلك . ويتم تخزين الرقم الممثل لعدد السجلات فى الملف المؤقت (Tempinv) زائدا واحد في متغير الذاكرة (StartTrans). وذلك لكى تبدأ الفاتورة التالية من هذا الرقم عندما يريد المستخدم ارسال فاتورة أخرى باسم بائع آخر (Vendor) . والسطور التالية توضح هذه العملية .

```
IF Pinvoice "Y"
    SET PRINT ON
    ? "Date:" , T_Date
    ? "Invoice number:" , STR(Minvoice,5)
    ? "Customer:" , MCust , SPACE(20)
    ?? "Clerk:" , MClerk
    ? Uline
    ?
    SELECT B
    GOTO StartTrans
    LIST OFF WHILE .NOT. EOF() Part_no, A --> P_name,;
    Qty , Price, Qty * Price
    ?
    ?
    ? "Total:" , SPACE(34), Mtotal
    EJECT
    SET PRINT OFF
    StartTrans = RECCOUNT() + 1
ENDIF
```

وفى الجزء التالى يتم سؤال المستخدم اذا كان يريد طباعة فواتير أخرى . فاذا أراد ذلك يتم تنفيذ الحلقة مرة ثانية حيث يكون المتغير (Again) مخزنا فيه القيمة ("Y") . أما اذا أراد الخروج فانه يكتب (N) . وفى هذه الحالة يتوقف تنفيذ الحلقة . ويتم ذلك من خلال السطور التالية :

```
CLEAR
@ 23,2 SAY "Do another transaction? (Y/N)" ;
  GET Again PICT "!"
```

```
add invoices while user does not request exit)
```

وفى الجزء التالى يقوم البرنامج باضافة السجلات الموجودة ف
الوقت (Tempinv) الى آخر ملف المبيعات ، كمـا يتم تحد
الفهرس . ويقوم البرنامج أيضا بمسح السجلات الموجودة فى الم
لتوفيـر المساحة التخزينية على القرص . ثم يقوم البرنامج باغ
الملفات المفتوحة والعودة الى البرنامج الرئيسي لتشغيل برنامج ا
(SMenu.prg) . ويتم ذلك من خلال السطور التالية :

```
DATABASES

ating transaction file , please wait ..."
LK ON
LES
 FROM Tempinv
mpinv

Les INDEX Sales
C
LK OFF
DATABASES
```

١٥ - ٣ - ٣ ادخال السعر آليا

يجدر الاشارة هنا الى أن هذا البرنامج عام ، يمكن استخداه
نقطة بيع . لذلك فانه يبدو طويلا ومعقدا بعض الشيء . كما يمك
ليناسب الظروف والمطالب المختلفة للمستخدم . حيث يمكن م
السعـر آليا فى فاتورة البيع (Invoice) بدلا من كتابتـ
المستخدم . ولاجراء هذا التعديل يلزم أولا اضافة حقل سـ
(Price) فى الملف الرئيسى (Master.dbf) ثم تعديل
بناء على ذلك .

١٥ - ٤ برنامج تقارير البيع

يتم تنفيذ هذا البرنامج عندما يختار المستخدم الرقم (2) من قائمة تشغيل ملف المبيعات (SMenu) . وهذا البرنامج يتيح للمستخدم عرض أو طباعة تقارير اما لفاتورة محددة أو لحركة المبيعات خلال فترة محددة (أى من تاريخ محدد الى تاريخ آخر) . وعند تشغيل هذا البرنامج تظهر الشاشة التالية :

```
Sales Report Options          02/20/90      09:30:25

1.  By invoice number
2.  By dates
3.  Return to sales menu

Enter choice (1-3)
```

شكل (١٥ - ٩)

وعندما يختار المستخدم الرقم (١) من قائمة برنامج تقارير مبيعات الأصناف لعرض أو طباعة تقرير عن فاتورة محددة يظهر السؤال التالى :

Look for what invoice number?

وعندما يقوم المستخدم بادخال رقم الفاتورة (Invoice number) يظهر التقرير الخاص بهذه الفاتورة على الشاشة أو على الطابعة كالآتى مثلا :

Invoice number : 1246 Date : 02/20/90
Clerk : Hassan Customer: D. FATHY

Part #	Qty	Part name	Price	Total
AAA	1	printer	900	900
BBB	10	floppy disk	16	160

شكل (١٥ - ١٠)

وعندما يختار المستخدم الرقم (2) من القائمة لعرض بيانات حركة البيع خلال فترة محددة ، يظهر الآتى على الشاشة :

```
Enter start date     :    /    /
Enter end date       :    /    /
```

وعندما يكتب المستخدم تاريخ البداية وتاريخ النهاية يظهر التقرير الخاص بحركة المبيعات خلال هذه الفترة على الشاشة أو على الطابعة كالآتى :

Page No. 1
02/20/90

Sales Transactions

Date	Invoice	Sales person	Customer	Part	Qty	Sale	Price
02/01/90	1243	Magdy	Salem	SAF	20	700	
02/10/90	1368	Medhat	Shawky	LAM	10	500	
* * TOTAL * *						1200	

شكل (١٥ - ١١)

ويمكن انشاء التقرير باستخدام الأمر (CREATE) أو الأمر (MODIFY) وذلك كالآتى :

```
CREATE REPORT Sales
```

على أن تكون محتويات الأعمدة كالآتى :

Column	Contents	Heading	Width	Decimals	Total?
1	Date	Date	9		
2	Invoice_no	Invoice	7		
3	Clerk	Salesperson	14		
4	Customer	Customer	14		
5	Part_no	Part	6		
6	Qty	Qty	5	0	N
7	Cost	Sale Price	10	0	Y

شكل (١٥ - ١٢)

١٥ - ٤ - ١ كتابة الخطوات الأولية للبرنامج

كما سبق الايضاح فان الوظيفة الرئيسية لبرنامج تقارير المبيعات (Salreps.prg) هى سؤال المستخدم عن نوع التقرير الذى يريده . ثم عرض هذا التقرير على الشاشة أو طباعته على الطابعة . والخطوات الأولية لهذا البرنامج كالآتى :

١ - يتم فتح ملف المبيعات (Sales.dbf) والملف الرئيسى (Master.dbf).
٢ - يتم ربط الملفين .
٣- يتم تكوين حلقة تكرارية لعرض قائمة الاختيارات .
٤- يتم سؤال المستخدم عن نوع التقرير المطلوب .
٥- فى حالة طلب التقرير بناء على الفاتورة يتم سؤال المستخدم عن رقم الفاتورة المطلوبة .
٦- يتم البحث عن أول سجل يحتوى على هذا الرقم فى ملف المبيعات (Sales.dbf) .
٧- عند العثور على هذا السجل يتم طباعة رأس التقرير بناء على البيانات الموجودة فى هذا السجل .
٨ - يتم طباعة بيانات الأصناف الخاصة برقم الفاتورة المطلوبة .
٩ - فى حالة طلب التقارير خلال فترة زمنية محددة يتم سؤال المستخدم عن تاريخ البداية وتاريخ النهاية .
١٠- يتم تحويل التواريخ التى يدخلها المستخدم من الصورة الحرفية الى الصورة التاريخية .
١١- يتم طباعة التقارير خلال هذه الفترة باستخدام صورة التقرير (Sales.frm) .
١٢- عند الانتهاء يتم الرجوع الى القائمة الرئيسية لبرنامج المبيعات (SMenu.prg) .

١٥ - ٤ - ٢ كتابة البرنامج

يتكون برنامج تقارير البيع من السطور التالية :

```
* * * * * * * * * * * * * * * *   * SalReps.prg
*      Print reports from the sales file
*      Called from sales menu, SMenu.prg
```

```
* - - - - -  Open files and set up relationship.
SELECT 1
USE Sales
SELECT 2
USE Master INDEX Master

SELECT 1
SET RELATION TO Part_no INTO Master
Repchoice = 0

* - - - - - -  Start loop for menu
DO WHILE Repchoice # 3
    CLEAR
    @ 2,1  SAY "Sales Report Options"
    @ 2,60 SAY DTOC(T_Date) + "    " + TIME()
    @ 3,0  SAY Uline
    ?
    ?
    TEXT
        1. By invoice number
        2. By dates
        3. Return to Sales menu
    ENDTEXT
    @ 24,1 SAY "Enter choice(1-3)" ;
      GET Repchoice PICT "9" RANGE 1,3
    READ

    * - - - - - -  IF not exiting , ask about printer.
    @ 5,0  CLEAR
    STORE "  " TO YN, Printer
    IF Repchoice # 3
        @ 15,5 SAY "Send to printer " GET YN PICT "!"
        READ
        * - - - - - - Set up printer macro
        IF YN = "Y"
            Printer = "TO PRINT"
        ENDIF
     ENDIF
     * - - -Print appropriate report based on request.
```

```
@ 5,0  CLEAR
DO CASE
    * - - - - Case 1 : Search by invoice number.
    CASE Repchoice = 1
       @ 15,5
       INPUT "Look for what invoice?" TO Isearch
       CLEAR
       LOCATE FOR invoice_no = Isearch
       * - - - If found , print invoice.
       IF FOUND()
          IF YN = "Y"
            SET PRINT ON
          ENDIF
          * - -  Print header from first record
          * - - -  with that invoice number.
          ? "Invoice number : " , Invoice_no
          ?? "Date:" , Date
          ? "Clerk:", Clerk,"Customer:", Customer
          ?
          ? "Part #  Part name  Qty Price Total "

          * - - - Print data for all records with
          * - - - that invoice number.
          LIST OFF WHILE Invoice_no = Isearch ;
            Part_no,B-->P_name, Qty , Price , ;
            (Qty * Price)
          IF YN = "Y"
              EJECT
              SET PRINT OFF
          ENDIF
       ENDIF(found)

    * - - - - - -  Case 2 : Search by dates
    CASE Repchoice = 2
       STORE SPACE(8) TO Start , End
       @ 15,5 SAY "Enter start date" GET Start ;
         PICT " 99/99/99"
       @ 17,5 SAY "Enter end date" ;
```

```
     GET End PICT "99/99/99"
    READ
    Start = CTOD(Start)
    End = CTOD(End)
    CLEAR
    * - - - - -  Print the report.
    REPORT FORM Sales FOR Date >= Start .AND.;
      Date <= End & Printer
\SE

- - IF not going to printer, Pause the screen
I # "Y" .AND. Repchoice # 3
)
)
/AIT "Press any key to return to the reports ;
  menu"
F
.le user does not request to exit)

- -  When done return to sales menu
'ION TO
'ABASES
```

والجزء الأول من البرنامج يبدأ كالمعتاد باسـ (Salreps.prg) ووظيفته والبرنامج القائم باستدعائه ، ثـ اللفات . ويلاحظ هنا أن ملف المبيعات (Sales.dbf) يـ فتح الفهرس معه . وذلك لأن المطلوب طباعة تقارير تعتمد على (Ivoice number) أو التاريخ (Date) . وحيث أن تخزينها فعليا فى ملف المبيعات مرتبة حسب رقم الفاتورة وتا لـذلك فان الملف يكون مرتبا بالترتيب المطلوب دون الحاجـ الفهرس . ولادخال اسم الصنف فى الفاتورة يلزم فتح الـ (Master.dbf) وربطه بملف المبيعات . والسطور التاليـ الخطوات .

```
 * * * * * * * * * *  SalReps.prg
ed from sales menu , SMenu.prg
 -  Open files and set up relationship.
```

```
SELECT 1
USE Sales
SELECT 2
USE Master INDEX Master
SELECT 1
SET RELATION TO Part_no INTO Master
```

والجزء الثانى يتم من خلاله تكوين حلقة تكرارية لعرض قائمة الاختيارات على المستخدم وسؤاله عن الاختيار المطلوب ثم تخزين هذا الاختيار فى متغير الذاكرة (Repchoice) . والسطور التالية توضح هذه الخطوات :

```
Repchoice = 0
* - - - - -  Start loop for menu
DO WHILE Repchoice # 3
    CLEAR
    @ 2,1  SAY "Sales Report Options"
    @ 2,60 SAY DTOC(T_Date) + "    " + TIME()
    @ 3,0  SAY Uline
    ?
    ?
    TEXT

         1. By invoice number
         2. By dates
         3. Return to Sales menu

    ENDTEXT
    @ 24,1 SAY "Enter choice(1-3)" ;
      GET Repchoice PICT "9" RANGE 1,3
    READ
```

والجـــزء الثالث يوضح انشاء متغير الذاكرة (Printer) وادخال القيمة (TO PRINT) فى هذا المتغير لاستخدامه كماكرو لتشغيل الطابعة بعد ذلك ، وذلك فى حالة عدم اختيار المستخدم للرقم (3) من القائمة للخروج . ويتم ذلك من خلال السطور التالية :

```
* - - - - -IF not exiting , ask about printer.
@ 5,0 CLEAR
STORE "   " TO YN, Printer
IF Repchoice # 3
    @ 15,5 SAY "Send to printer " GET YN PICT "!"
    READ
    * - - - - - - Set up printer macro
    IF YN = "Y"
        Printer = "TO PRINT"
    ENDIF
ENDIF
```

والجزء الرابع يوضح استخدام الأمر (DO CASE) في طباعة التقرير المطلوب حسب اختيار المستخدم . فاذا أراد المستخدم الطباعة بناء على رقم الفاتورة ، فان البرنامج يسأل عن رقم الفاتورة المطلوب كما يحدد أول سجل يحتوى على هذا الرقم . ومن هذا السجل يقوم البرنامج بعرض عنوان التقرير (Heading) الذى يحتوى على اسم الموظف القائم بعملية البيع (Clerk) واسم العميل المشترى (Customer) وتاريخ البيع . ثم يستخدم الأمر (LIST) والعبارة (WHILE) لعرض أو طباعة بيانات جميع السجلات التى تشترك فى رقم الفاتورة . والسطور التالية توضح هذه العملية :

```
*  - - - - -Print appropriate report based on request.
@ 5,0  CLEAR
DO CASE

    * - - - - - -  Case 1 : Search by invoice number.
    CASE Repchoice = 1
    @ 15,5
    INPUT "Look for what invoice  ? " TO Isearch
    CLEAR
    LOCATE FOR invoice_no = Isearch
    * - - - - - -  If found , print invoice.
    IF FOUND()
        IF YN = "Y"
            SET PRINT ON
        ENDIF
        * - - - - - -  Print header from first record
```

```
* - - - - - -  with that invoice number.
? "Invoice number : " , Invoice_no
?? "Date:" , Date
? "Clerk:" , Clerk, "Customer:" , Customer
?
? "Part  #  Part name  Qty Price  Total "
* - - -  Print data for all records with that
* - - - - - -  invoice number.
LIST OFF WHILE Invoice_no = Isearch , ;
  Part_no,B-->P_name, Qty , Price , ;
  (Qty * Price)
IF YN = "Y"
    EJECT
    SET PRINT OFF
ENDIF
ENDIF(found)
```

والجزء الخامس يتم من خلاله عرض بيانات التقرير فى حالة طلب المستخدم تقريرا بالمبيعات التي تمت خلال فترة معينة . وفي هذه الحالة يسأل البرنامج عن تاريخ البداية (Start) وتاريخ النهاية (End) ثم يستخدم صورة التقرير (Sales.frm) التى سبق انشاؤها . ويتم ذلك من خلال السطور التالية :

```
* - - - - - -  Case 2 : Search by dates
CASE Repchoice = 2
    STORE SPACE(8) TO Start , End
    @ 15,5 SAY "Enter start date" GET Start ;
    PICT " 99/99/99"
    @ 17,5 SAY "Enter end date" ;
    GET End PICT "99/99/99"
    READ
    Start = CTOD(Start)
    End = CTOD(End)
    CLEAR
    * - - - - - -  Print the report.
    REPORT FORM Sales FOR Date >= Start .AND. ;
    Date <= End & Printer
ENDCASE
```

ويلاحظ هنا عملية تحويل التاريخ من الصورة الحرفية التى يدخلها المستخدم الى الصورة التاريخية التى يستطيع البرنامج التعامل معها ، وذلك باستخدام الدالة (CTOD) .

والجزء السادس يتم من خلاله ايقاف الشاشة لحظيا (Pause) في حالة عدم الرغبة في طباعة التقرير والاكتفاء بعرضه على الشاشة . ويتم ذلك من خلال السطور التالية :

```
* - - - - -IF not going to printer, Pause the screen
IF YN # "Y" .AND. Repchoice # 3
    ?
    ?
    WAIT "Press any key to return to the reports ;
    menu"
ENDIF
```

والجزء السابع يتم من خلاله انهاء الحلقة التكرارية واغلاق الملفات المفتوحة ثم العودة الى البرنامج القائم بالاستدعاء (SMenu.prg) . ويتم ذلك من خلال السطور التالية :

```
ENDDO(while user does not request to exit)
* - - - - - -  When done return to sales menu
SET RELATION TO
CLOSE DATABASES
RETURN
```

الفصل السادس عشر

برنامج تشغيل ملف الاضافة



هذا البرنامج هو جزء من برنامج المخازن (Inventory) يختص بتسجيل بيانات الأصناف التي يتم توريدها الى المخزن . ويتم تخزين بيانات هذه الأصناف في ملف الاضافة (Newstock.dbf) الذى سبق انشاؤه . ويتم تشغيل هذا البرنامج عندما يختار المستخدم الاختيار رقم (3) من القائمة الرئيسية لبرنامج المخازن . انظر الشكل (١٦ - ١) .

```
Inventory system main menu              02/20/90

      1. Manage master inventory
      2. Record sales
      3. Record new stock
      4. Exit

Enter choice   0
```

شكل (١٦ - ١)

```
New stock system menu      02/20/90   08:30:45

      1. Record new items
      2. Print new stock data
      3. Edit new stock data
      4. Return to main menu

Enter choice   ( 1 - 4 )
```

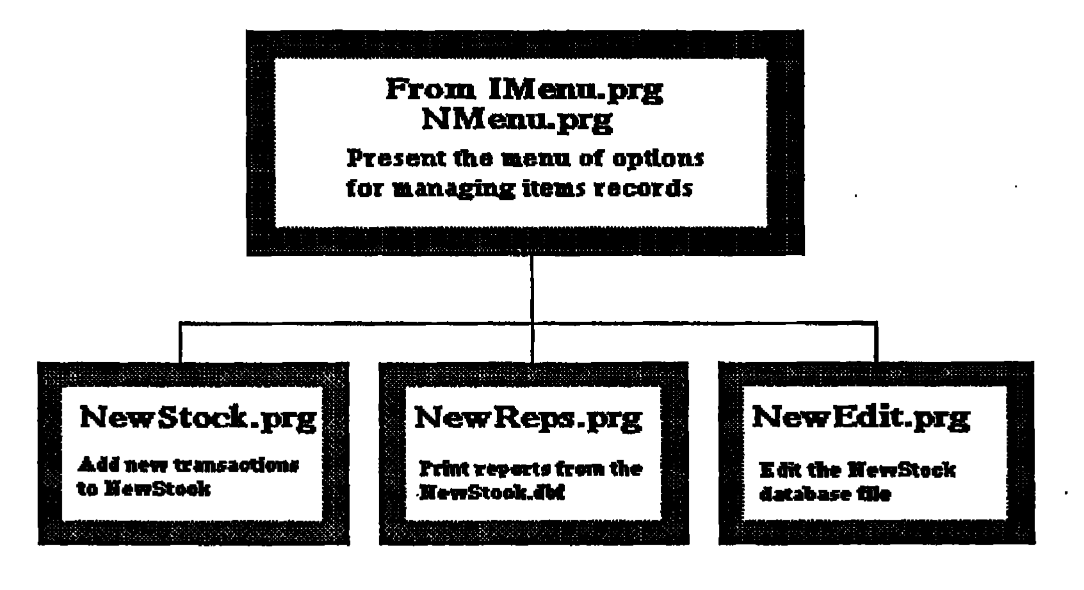
شكل (١٦ - ٢)

وفي هذه الحالة تظهر القائمة الخاصة بالاضافة . أنظر الشكل (١٦ - ٢) . وهذه القائمة تتيح للمستخدم اضافة حركة جديدة (Transaction) ، أو طباعة تقارير ، أو تصحيح بيانات الحركة ، أو الرجوع الى القائمة الرئيسية للمخازن .

١٦ - ١ تركيب البرنامج

يتكون برنامج الاضافة من أربعة برامج منفصلة . البرنامج الأول هو برنامج (NMenu.prg) الذي يقوم بالتحكم في تشغيل البرامج الثلاثة الأخرى . وبناء على اختيار المستخدم يتم التفرع الى البرنامج (NewStock.prg) الذى يسمح للمستخدم باضافة بيانات الأصناف الجديدة . أو يتم التفرع الى البرنامج (NewRep.prg) الذى يطبع التقارير . أو يتم التفرع الى البرنامج (NewEdit.prg) الذى يسمح بتعديل بيانات أى حركة اضافة .

والشكل التالى يوضح التركيب الهرمى للبرنامج .



شكل (١٦ - ٢)

١٦ - ٢ برنامج قائمة الاضافة (NMenu.prg)

وهذا البرنامج لايختلف عن برامج القوائم التى سبق شرحها لذلك يتم عرض سطور البرنامج دون الحاجة الى شرحها مرة ثانية وهى كالآتى :

```
* * * * * * * * * * * * * NMenu.prg
* Menuf or managing New Stock portion of Inventory
* system . Called from Inventory System main menu.
* - - - - - - -  Set up loop form presenting menu.
Nchoice = 0
DO WHILE NChoice # 4
     CLEAR
     @ 2,1  SAY "New Stock System menu"
     @ 2,60 SAY DTOC(T_Date) + "    " + TIME()
     @ 3,0  SAY Uline
     ?
     ?
     TEXT
            1. Record new items
            2. Print new stock reports
            3. Edit new stock data
            4. Return to main menu
    ENDTEXT
    @ 24,1 SAY "Enter choice(1-4)"
      GET Nchoice PICT "9" RANGE 1,4
    READ

    * - - - - - -  Branch to appropriate program.
    DO CASE
        CASE Nchoice = 1
            DO NewStock
        CASE Nchoice = 2
            DO NewReps
        CASE Nchoice = 3
            DO NewEdit
     ENDCASE
```

```
ENDDO(while Nchoice # 4)
* - - - - -  Return to main menu
RETURN
```

١٦ - ٣ برنامج ادخال بيانات الأصناف (Newstock.prg)

عندما يختار المستخدم الرقم (1) من قائمة الاضافة فان برنامج القائمة (NMenu.prg) ينتقل الى البرنامج (NewStock.prg) لينفذه . وفى هذه الحالة يتم مسح الشاشة وتظهر الرسالة التالية للمستخدم :

```
Enter data for goods received
Part number :-
```

وعندما يكتب المستخدم رقما غير موجود في قاعدة البيانات يتم تحذير المستخدم بصفارة (Beep) ثم تظهر رسالة توضح له عدم وجود هذا الجزء ، مع اعطائه الفرصة للمحاولة مرة ثانية . وذلك كالآتى :

```
Enter data for goods received
Part number :-          :No such part !!!
```

ويجب ملاحظة أن هذا البرنامج لايضيف أصناف بأرقام جديدة ، ولكنه يضيف كميات من أصناف موجودة أرقامها فى قاعدة البيانات . وهذا عكس البرنامج (Addnumbs.prg)المستخدم فى برنامج تشغيل الملف الرئيسى (Master.dbf) .

وعند ادخال المستخدم لرقم جزء موجود يقوم البرنامج بعرض اسم هذا الجزء على الشاشة بالاضافة الى تاريخ ادخال هذا الصنف (تاريخ اليوم الحالى) واسم البائع (Vendor) . كما يتيح له تعديل اسم البائع (Vendor) حسب الحاجة ثم يطلب من المستخدم كتابة الكمية الواردة من هذا الصنف وسعر الشراء . وذلك كالآتى :

```
Enter data for goods received
Part number  A-122          Shirt
Quantity : -                Price:
Date 02/20/90               Vendor : Hasan
```

وعند الانتهاء من ادخال البيانات في الحقول الخالية ، فان البرنامج يسأل عن الصنف التالى المطلوب اضافته . وعند الانتهاء من ادخال الأصناف ، يقوم المستخدم بالضغط على مفتاح الادخال للرجوع الى قائمة الاضافة مرة ثانية .

١٦ - ٤ كتابة الخطوات الأولية (PSEUDOCODE)

يتم كتابة الخطوات الأولية للبرنامج كالآتى :

١ - يتم فتح الملف الرئيسى (Master.dbf)وملف الاضافة (NewStock.dbf) .
٢ - يتم تكوين حلقة تكرارية لتسجيل بيانات الأصناف الجديدة .
٣ - عند ادخال رقم الجزء يقوم البرنامج بالبحث عن هذا الجزء فى الملف الرئيسى (Master.dbf).
٤ - عند عدم العثور على رقم الجزء يتم تحذير المستخدم والسماح له بادخال رقم جديد .
٥ - عند العثور على رقم الجزء يتم عرض بياناته وسؤال المستخدم عن الكمية الواردة وسعر الشراء .
٦ - يتم الاستمرار في ادخال الأصناف حتى يطلب المستخدم الخروج .
٧ - يتم العودة الى قائمة الأضافة .

١٦ - ٥ كتابة البرنامج

يتم كتابة هذا البرنامج كالآتى :

```
* * * * * * * * * * * * * * * * * * * NewStock.prg
*    Data entry program for goods received
*     Called from New Stock, NMenu.prg
SELECT A
USE Master INDEX Master
SELECT B
USE NewStock INDEX NewStock
SELECT A

* - - - - - Set up loop for recording goods received.
Partnumb = "x"
DO WHILE Partnumb # "  "
     @ 5,0 CLEAR
```

```
 Partnumb = SPACE(5)
@ 10,2 SAY "Enter data for goods received"
@ 12,4 SAY "Partnumber" GET Partnumb
READ
* - If a part number was entered , find it in
* - Master file
IF Partnumb # " "
     SEEK Partnumb
     DO CASE

         * - - -  If part not found, warn user and
         * - - - - - -  try again
         CASE .NOT. FOUND()
            @ 12,23 SAY "No such part !!!"
            ? CHR(7)
            * - If found Append a new record to
            * - NewStock .dbf, and get
            * - rest of data.
         CASE FOUND()
            @ 12,25 SAY P_name
            SELECT B
            APPEND BLANK
            REPLACE Part_no WITH Partnumb
            REPLACE Date WITH T_Date
            REPLACE Vendor WITH A -> Vendor
            @ 14,2 SAY "Quantity" GET Qty
            @ 14,22 SAY "Price" GET COST PICT ;
                   "99999.99"
            @ 16,2  SAY "Date" GET Date PICT ;
                   "99/99/99"
            @ 16,22 SAY "Vendor" GET Vendor
            READ
            SELECT A
       ENDCASE
  ENDIF(Partnumb# "  ")
ENDDO(While user does not want to quit)
```

```
* - - - - - -  Close files and return to New Stock
* - - - - - -  menu.
CLOSE DATABASES
RETURN
```

والجزء الأول من البرنامج يقوم بفتح الملف الرئيسي (Master.dbf) لاختبار رقم الجزء الذي يتم ادخاله . كما يتم فتح ملف الاضافة (NewStock.dbf) لتخزين بيانات الأصناف الجديدة . ويتم ذلك من خلال السطور التالية :

```
SELECT A
USE Master INDEX Master
SELECT B
USE NewStock INDEX NewStock
SELECT A
```

والجزء الثانى يقوم بتكوين حلقة تكرارية لادخال رقم الجزء . وذلك من خلال السطور التالية :

```
Partnumb = "x"
DO WHILE Partnumb # "  "
    @ 5,0 CLEAR
    Partnumb = SPACE(5)
    @ 10,2 SAY "Enter data for goods received"
    @ 12,4 SAY "Partnumber" GET Partnumb
    READ
```

والجزء الثالث يتم من خلاله البحث عن الرقم الذى قام المستخدم بادخاله وذلك من خلال السطور التالية :

```
IF Partnumb # " "
     SEEK Partnumb
```

والجزء الرابع يتم من خلاله تحذير المستخدم فى حالة ادخال رقم جزء غير موجود . وذلك من خلال السطور التالية :

```
DO CASE
     CASE .NOT. FOUND()
          @ 12,23 SAY "No such part !!!"
          ? CHR(7)
```

والجزء الخامس يتم من خلاله عرض اسم الصنف الذى يتم ادخال رقمه على الشاشة ، وذلك فى حالة ادخال المستخدم لرقم موجود . كما يتم اضافة سجل خال فى نهاية ملف الاضافة (NewStock.dbf) حتى يتم ادخال البيانات الجديدة فيه . كما يتم ملء بيانات رقم الصنف والتاريخ واسم البائع آليا وسؤال المستخدم عن الكمية والسعر واسم البائع فى حالة تغييره . ثم يتم التجهيز لادخال صنف جديد . ويتم ذلك من خلال السطور التالية :

```
CASE FOUND()
        @ 12,25 SAY P_name
        SELECT B
        APPEND BLANK
        REPLACE Part_no WITH Partnumb
        REPLACE Date WITH T_Date
        REPLACE Vendor WITH A -> Vendor
        @ 14,2 SAY "Quantity" GET Qty
        @ 14,22 SAY "Price" GET COST PICT "99999.99"
        @ 16,2  SAY "Date" GET Date PICT "99/99/99"
        @ 16,22 SAY "Vendor" GET Vendor
        READ
        SELECT A
```

والجزء السادس يتم من خلاله اغلاق الملفات بعد انتهاء الحلقة التكرارية ثم العودة الى قائمة الاضافة . وذلك من خلال السطور التالية :

```
          ENDCASE
     ENDIF(Partnumb# "  "
ENDDO(While user does not ask to quit)
CLOSE DATABASES
RETURN
```

١٦ - ٦ برنامج تقارير الاضافة (NewReps.prg)

يسمح هذا البرنامج للمستخدم بمراجعة بيانات الأصناف الجديدة فى صورة تقارير بناء على رقم الجزء (Part number) أو خلال فترة زمنيه محددة . وتفيد هذه التقارير في حل أى خلافات قد تنشأ مع البائعين (Vendors) عن طريق امدادهم بالبيانات الدقيقة عن الأصناف وأسعارها .

ويتم تشغيل هذا البرنامج عندما يختار المستخدم الرقم (2) من قائمة الاضافة . انظر الشكل (١٦ - ٤) .

```
New Stock System Menu   02/20/90   02:20:45

        1. Record new items
        2. Print new stock data
        3. Edit new stock data
        4. Return to main

   Enter choice ( 1 - 4 )  0
```

شكل (١٦ - ٤)

فى هذه الحالة تظهر القائمة التالية على الشاشة .

```
1.    By part number
2.    By dates
3.    Return to New Stock menu
```

وعند اختيار المستخدم للرقم (1) يظهر سؤال آخر عما اذا كان المطلوب طباعة التقرير أم عرضه فقط على الشاشة . ثم يظهر سؤال آخر عن رقم الجزء المطلوب عرض أو طباعة التقرير له . وعند كتابة المستخدم لهذا الرقم يظهر تقرير يوضح حركة الاضافة لهذا الصنف .

وعند اختيار المستخدم للرقم (2) من القائمة ، فان هذا يعنى أنه يريد عرض حركة الأصناف من تاريخ معين الى تاريخ آخر . وفى هذه الحالة يظهر على الشاشة سؤال عن تاريخ البداية وتاريخ النهاية كالآتى :

```
Enter start date :   /   /
Enter end date   :  /    /
```

فيقوم المستخدم بكتابة المطلوب .وفى هذه الحالة تظهر كل بيانات حركة الاضافة التى تمت بين هذين التاريخين .

أنظر الشكل (١٦ - ٥) ، الشكل (١٦ - ٦)

Page No. 1					02/20/90
Inventory Items Received					
Part Name	Part	Qty	Purchase Price	Date	Vendor Name
A-122	Shirt	20	500	02/20/90	Hasan
A-122	Shirt	35	750	02/01/90	Salem

الشكل(١٦ - ٥)

Page No. 1					02/20/90
		Inventory Items Received			
Part	Part Name	Qty	Purchase Price	Date	Vendor Name
A-122	Shirt	30	750	02/01/90	Hasan
A-122	Shirt	20	500	02/20/90	Salem
A-121	Shoes	20	500	02/25/90	Hytham

شكل (١٦ - ٦)

ولكتابة البرنامج يتم أولا انشاء صورة التقرير المطلوب . مع ملاحظة أنه يمكن انشاء صورة واحدة للتقرير واستخدامها في حالة طلب التقرير بناء على رقم الجزء أو بناء على فترة زمنية محددة . ويتم ذلك باستخدام الأمر (CREATE) أو الأمر (MODIFY) . وحيث أن التقرير يجب أن يتضمن بيانات من ملف الاضافة (NewStock.dbf) ، وكذلك بيانات من الملف الرئيسي (Master.dbf) ، لذلك يلزم عند انشاء التقرير فتح ملف الاضافة وكذلك الملف الرئيسي والربط بينهما . ويتم ذلك من خلال الأوامر التالية :

```
CLEAR ALL
SELECT 1
USE NewStock
SELECT 2
USE Master INDEX Master
SELECT 1
SET RELATION TO Part_no INTO Master
MODIFY REPORT NewStock
```

مع ملاحظة أن هذه الأوامر تكتب من مشيرة النقطة (Dot Prompt) قبل بداية كتابة البرنامج .

وعندما تظهر الشاشة الخاصة بانشاء التقرير يتم ادخال محتويات أعمدة التقرير (Columns) كما يظهر من الشكل التالى :

Column	Contents	Heading	Width	Decimals	Total?
1	Part_no	Part	6		
2	B->P_name	Part Name	15		
3	Qty	Qty	4	2	N
4	Cost	Purchase price	12	2	N
5	Date	Date	8		
6	Vendor	Vendor	25		

شكل (١٦ - ٧)

ويلاحظ هنا كتابة (B ->P_name) للحصول على اسم الجزء من الملف الرئيسى (Master.dbf) .

١٦ - ٧ كتابة البرنامج

نظرا لأن هذا البرنامج شبيه ببرامج التقارير التى سبق كتابتها مع الملف الرئيسي (Master.dbf) وملف المبيعات (Sales.dbf) ، فسوف يكتفى هنا بكتابة البرنامج فقط ويمكن الرجوع الى برامج التقارير السابقة لمتابعة شرحها . والبرنامج يتكون من السطور التالية :

```
* * * * * * * * * * * * * * *NewReps.prg
*    Print reports from the NewStock file
*    Called from NewStock menu, NMenu.prg .

* - - - - -  Open NewStock and Master databases.
SELECT 1
USE NewStock
```

```
SELECT 2
USE Master INDEX Master
* - - - - -  Set up relationship.
SELECT 1
SET RELATION TO Part_no INTO Master

* - - - - -  Start loop for menu
Repchoice = 0
DO WHILE Repchoice # 3
   CLEAR
   @ 2,1  SAY "New Stock Report options"
   @ 2,60 SAY DTOC(T_Date) + "   " + TIME()
   @ 3,0  SAY Uline
   ?
   ?
   TEXT
      1. By part number
      2. By dates
      3. Return to New Stock menu
   ENDTXT
   @ 24,1 SAY "Enter choice(1-3)" ;
     GET Repchoice PICT "9"
   READ

   * - - - - -  If not exiting , ask about printer.
   @ 3,0  CLEAR
   STORE "  "  TO YN, Printer
   IF Repchoice # 3
      @ 15,5 SAY "Send to printer?" GET YN PICT "!"
      READ
      * - - - - -  Set up printer macro.
      IF YN = "Y"
         Printer = "TO PRINT"
      ENDIF
   ENDIF
```

```
   * - - - - -  If not going to pritner, pause the
   * - - - - -  screen.
   IF YN # "Y" .AND. Repchoice # 3
      ?
      ?
      WAIT "Press any key to return to the Repchoice;
                    menu"
   ENDIF
ENDDO (while user does not request to exit)
* - - - - -  when done , return to New Stock menu
SET RELATION TO
CLOSE DATABASES
RETURN
```

الفصل السابع عشر

برنامج تحديث البيانات



فى الأبواب السابقة تم انشاء ثلاثة أجزاء رئيسية من نظام المخازن للتحكم فى الملف الرئيسى (Master.dbf) وملف المبيعات (Sales.dbf) وملف الاضافة (NewStock.dbf) . وفى هذا الباب يتم انشاء البرنامج المكمل للنظام الذى يمثل أهم جزء فيه . وهذا البرنامج يقوم بتحديث (Updating) لبيانات الملف الرئيسى (Master.dbf) بناء على البيانات الموجودة فى ملف المبيعات (Sales.dbf) وملف الاضافة (NewStock.dbf) . وفى هذا الباب أيضا يتم انشاء برنامج تعديل ملف المبيعات وبرنامج تعديل ملف الاضافة .

١٧ - ١ برنامج تحديث الملف الرئيسى (Master.dbf)

يقوم هذا البرنامج بخصم كميات الأصناف الموجودة فى ملف المبيعات من الكميات الموجودة فى الملف الرئيسى لكل صنف تم البيع منه . كما يقوم باضافة كميات الأصناف الموجودة فى ملف الاضافة الى الكميات الموجودة فى الملف الرئيسى .

ويقوم هذا البرنامج أيضا بخصم الكميات الموجودة فى ملف الاضافة من الكميات الموجودة فى حقل تحت الطلب (On_order) فى الملف الرئيسى ، حيث أن هذه الأصناف تم اضافتها بالفعل . كما يقوم بتعديل سعر الشراء لهذه الأصناف (Purchase Price) بالسعر الموجود فى ملف الاضافة ، حيث أن هذا السعر يعتبر أحدث سعر للصنف . وأخيرا يقوم البرنامج بتعديل تاريخ آخر تحديث للبيانات بالتاريخ الموجود فى ملف الحركة المستخدم .

ويتم تشغيل هذا البرنامج عندما يختار المستخدم الرقم (4) من القائمة الرئيسية لبرنامج تشغيل الملف الرئيسى (Master.dbf) . انظر الشكل (١٧ - ١)

```
Manage Master Inventory     02/20/90        12:40:50

          1. Add new part numbers
          2. Print reports
          3. Make changes
          4. Update from sales and Newstock
          5. Return to main menu

Enter choice (1 - 5)
```

شكل (١٧ - ١)

ولكتابة هذا البرنامج يتم أولا كتابة الخطوات الأولية (PSEUDOCODE) .

١٧ - ١ - ١ كتابة الخطوات الأولية (PSEUDOCODE)

١ - يتم مسح الشاشة .
٢ - يتم عرض رسالة توضح للمستخدم أن الملف الرئيسى جارى تحديثه من ملف المبيعات .
٣ - يتم فتح ملف المبيعات والفهرس الخاص به .
٤ - يتم نسخ جميع سجلات ملف المبيعات التى لم يتم تحديثها (لم يتم ترحيل بياناتها) الى ملف مؤقت (Temporary file) .
٥ - يتم فتح الملف المؤقت .
٦ - يتم التأكد من وجود سجلات فى الملف المؤقت (التأكد أن الملف المؤقت ليس فارغا) .
٧ - فى حالة وجود سجلات فى الملف المؤقت ، يتم فتح الملف الرئيسى وملف الفهرس الخاص به .
٨ - يتم تحديث الملف الرئيسى من الملف المؤقت وذلك بخصم كميات الأصناف المباعة وتعديل حقل تاريخ آخر تحديث .
٩ - يتم اغلاق جميع الملفات .
١٠- يتم فتح ملف المبيعات (Sales.dbf) .
١١- يتم ادخال القيمة (.T.) أى صحيح فى حقل الترحيل (Posted) لجميع سجلات ملف المبيعات . لأن هذا يوضح أن جميع السجلات قد تم ترحيلها (Posted) .
١٢- يتم اغلاق جميع الملفات .
١٣- يتم عرض رسالة للمستخدم توضح أن الملف الرئيسى جارى تحديثه من ملف الاضافة (NewStock.dbf) .
١٤- يتم فتح ملف الاضافة وملف الفهرس الخاص به .
١٥- يتم نسخ جميع سجلات ملف الاضافة التى لم يتم تحديثها (لم يتم ترحيل بياناتها) الى ملف مؤقت (Temporary file) .
١٦- يتم فتح الملف المؤقت .
١٧- يتم التأكد من وجود سجلات فى الملف المؤقت .
١٨- فى حالة وجود سجلات فى الملف المؤقت ، يتم فتح الملف الرئيسى وملف الفهرس الخاص به .
١٩- يتم تحديث الملف الرئيسى من الملف المؤقت وذلك باضافة كميات الأصناف المضافة وتعديل سعر الصنف وتاريخ آخر تحديث له وكذلك طرح كمية الصنف المضافة على المخزن من الكمية الموجودة فى حقل تحت الطلب (On_order) .

٢٠- يتم اغلاق جميع الملفات .
٢١- يتم فتح ملف الاضافة (NewStock.dbf) .
٢٢- يتم ادخال القيمة (.T.) أى صحيح فى حقل الترحيل (Posted) لجميع سجلات ملف الاضافة . حيث أن هذا يوضح أن جميع السجلات تم ترحيلها . و ذلك حتى لايعاد ترحيلها مرة ثانية .
٢٣- يتم اغلاق جميع الملفات .
٢٤- يتم الرجـــوع الى قائمـــة برنامج تشغيل الملف الرئيسى (MMenu.prg) .

١٧ - ١ - ٢ كتابة البرنامج

هذا البرنامج يتم تسميته (Updater.prg) ويتكـون من السطور التالية :

```
* * * * * * * * * * * * * * * * Updater.prg
*    Update the Master File from Sales and NewStock.
*    Called from Master menu , MMenu.prg.

* - - - - -  Ask user if sure abour updating.
YesNo = "Y"
@ 5,0  CLEAR
@ 15,4 SAY "Update Master file from Sales and" + ;
   "NewStock?(Y/N)" GET YesNo PICT "!"
READ
IF YesNo = "N"
    RETURN
ENDIF

* - - - - -  Display a message that Master is being
* - - - - -  updated from Sales file.
@ 5,0  CLEAR
@ 15,5 SAY "Updating from the Sales file ..."
USE Sales INDEX Sales

* - - - - -  Copy all unupdated records to Temp file.
COPY STRUCTURE TO Temp
COPY TO Temp FOR .NOT. Posted
```

```
* - - - - Make sure there are records in Temp
SELECT 2
USE Temp
IF RECCOUNT()>0
    * Use Master file and index for updating
    SELECT 1
    USE Master INDEX Master

    * - - - - - -  Update from the temporary sales file.
    UPDATE ON Part_no FROM Temp REPLACE Qty WITH ;
      Qty - Temp -> Qty , Date WITH Temp -> Date

    * - - - - - -  USE original Sales file , make all
    * - - - - - -  posted fields "True".
    CLOSE DATABASES
    USE Sales
    REPLACE ALL Posted WITH .T.
ENDIF(record count >0)
CLOSE DATABASES

* - - - - - -  Display a message that Master is bieng
* - - - - - -  updated from the NewStock file.
@ 15,5 SAY "Updating from the NewStock file"
USE NewStock INDEX NewStock
* - - - - Copy all unupdated records to Temp file.
COPY STRUCTURE TO Temp
COPY TO Temp FOR .NOT. Posted

* - - - - - -  Make sure there are records in temp.
SELECT 2
USE Temp
IF RECCOUNT() > 0
    * - - - USE Master file and index for updating
    SELECT 1
    USE Master INDEX Master
    * - - - Update from the temporary NewStock file.
    UPDATE ON Part_no FROM Temp ;
    REPLACE Qty WITH Qty + Temp -> Qty , Date WITH ;
```

```
    Temp -> Date, Cost WITH Temp -> Cost, ;
    On_order WITH On_order - Temp -> Qty

    * - - USE original NewStock file, make all posted
    * - - - - -  fields "True"
    CLOSE DATABASES
    USE NewStock
    REPLACE ALL Posted WITH .T.
ENDIF(record count > 0)

* - - - free up all work areas , and return to Master
* - - - menu.
CLOSE DATABASES
RETURN
```

والجزء الأول من البرنامج يبدأ بالتعريف باسم البرنامج (Updater.prg) ووظيفته ثم اسم البرنامج الذى قام باستدعائه (MMenu.prg) . ثم يقوم بمسح الشاشة وعرض رسالة للمستخدم للتاكد من رغبته فى التحديث (Updating) . ثم يتم عرض رسالة توضح للمستخدم أن الملف الرئيسى (Master.dbf) جارى تحديثه من ملف المبيعات . ثم يقوم البرنامج بفتح ملف المبيعات (Sales.dbf) وملف الفهرس الخاص به . ثم يقوم بنسخ السجلات التي لم يتم ترحيلها الى ملف مؤقت (Temp.dbf) . ويتم تنفيذ هذه الخطوات من خلال السطور التالية :

```
YesNo = "Y"
@ 5,0  CLEAR
@ 15,4 SAY "Update Master file from Sales and " + ;
"NewStock?(Y/N)" GET YesNo PICT "!"
READ
IF YesNo = "N"
   RETURN
ENDIF
* - - - - - -  Display a message that Master is being
* - - - - - -  updated from Sales file.
@ 5,0  CLEAR
@ 15,5 SAY "Updating from the Sales file ..."
USE Sales INDEX Sales
```

```
* - - - - - -  Copy all unupdated records to Temp file.
COPY STRUCTURE TO Temp
COPY TO Temp FOR .NOT. Posted
```

والجـــزء الثانى من البـرنامـج يقـوم بفتـح الملـف المؤقت للمبيعـات (Temp.dbf) الذى يحتوى على كل السجلات المطلوب ترحيل بياناتها الى الملف الرئيسى . ثم يقوم بالتأكد من وجود سجلات فى هذا الملف وذلك لتجنب التعامل مع ملف قاعدة بيانات فارغ ، لأن ذلك يمكن أن يؤدى الى نتائج غير متوقعة . ثم يتم فتح الملف الرئيسى (Master.dbf) والفهرس الخاص به فى منطقة عمل أخرى . ويتم ذلك من خلال السطور التالية :

```
*  Make sure there are records in Temp file.
SELECT 2
USE Temp
IF RECCOUNT()>0
    * Use Master file and index for updating
    SELECT 1
    USE Master INDEX Master
```

والجزء الثالث من البرنامج يتم من خلاله اجراء عمليات الترحيل بخصم الكمبيات الموجودة فى الملف المؤقت (Temp -> Qty) من الكميات الموجودة فى الملف الرئيسى (Qty) ثم يتم تعديل تاريخ آخر تحديث للملف الرئيسى (Date) بالتاريخ الموجود فى ملف المبيعات . ويتم ذلك من خلال السطور التالية :

```
* - - - - - - Update from the temporary sales file.
UPDATE ON Part_no FROM Temp REPLACE Qty WITH ;
Qty - Temp -> Qty , Date WITH Temp -> Date
```

والجزء الرابع يتم من خلاله تمييز السجلات التى تم ترحيلها حتى لايتم ترحيلها مرة ثانية . ويستخدم حقـــل الترحيل (Posted) فى هذه العمليـة . حيث يتم ادخال القيمة (.T.) أى (True) صحيح فى هذا الحقل لجـميـع سجلات ملف المبيعات . وهذا يعنى أن جميع البيانات قد تم ترحيلـها الى الملف الرئيسى . والسطور التالية توضح هذه العملية .

```
Close DATABASES
USE Sales
```

```
REPLACE ALL Posted WITH .T.
```

ثم يتم اغلاق جميع الملفات حتي يمكن البدء في تحديث الملف الرئيسى من ملف الأضافة . ويتم ذلك من خلال السطور التالية :

```
ENDIF(record count > 0)
CLOSE DATABASES
```

والجزء الخامس يتم من خلاله تنفيذ عملية التحديث من خلال ملف الاضافة (NewStock.dbf) بنفس الطريقة السابقة ، حيث يتم أولا عرض رسالة على المستخدم توضح له اجراء عملية التحديث ثم يتم فتح ملف الاضافة والفهرس الخاص به . ثم يتم نسخ جميع السجلات التي لم يتم تحديثها (ترحيلها من ملف الاضافة) الي ملف مؤقت (Temp) . وحيث أنه قد سبق كتابة الأمر (SET SAFETY OFF) في البرنامج الرئيسى لبرنامج المخازن (IMenu.prg) ، فان عملية نسخ السجلات تتم دون ظهور الرسالة التحذيرية المعتادة فى حالة النسخ فوق ملف سابق . والسطور التالية توضح تنفيذ هذه الخطوات .

```
@ 15,5 SAY "Updating from the NewStock file"
USE NewStock INDEX NewStock
* - - - - - Copy unupdated records to Temp file.
COPY STRUCTURE TO Temp
COPY TO Temp FOR .NOT. Posted
```

ثم يتم التأكد من وجود سجلات في الملف المؤقت (Temp) . وذلك من خلال السطور التالية :

```
SELECT 2
USE Temp
IF RECCOUNT() > 0
```

ثم يتم فتح الملف الرئيسى (Master.dbf) وملف الفهرس الخاص به من خلال السطور التالية :

```
SELECT 1
USE Master INDEX Master
```

ثم يقوم البرنامج بعد ذلك بترحيل القيم الموجودة في حقول الملف المؤقت (Temp) الى الملف الرئيسي (Master) . وذلك باضافة الكمية الموجودة في حقل الكمية (Temp -> Qty) على الكمية الموجودة في الملف الرئيسي (Qty) . كما يتم استبدال السعر بالسعر الموجود في الملف المؤقت (Temp->cost)وكذلك تاريخ آخر تحديث . كما يتم طرح الكمية (Temp->Qty) من الكمية الموجودة في حقل تحت الطلب (On_order) . والسطور التالية توضح تنفيذ هذه الخطوات .

```
* - - - - - Update from the temporary NewStock file.
UPDATE ON Part_no FROM Temp ;
REPLACE Qty WITH Qty + Temp -> Qty , Date WITH ;
Temp -> Date, Cost WITH Temp -> Cost, ;
On_order WITH On_order - Temp -> Qty
```

وفي الجزء الأخير من البرنامج يتم ادخال القيمة (.T.) في كل حقول الترحيل (Posted) حتى لايتم ترحيلها مرة ثانية . ثم يتم اغلاق جميع الملفات والعودة الى قائمة تشغيل الملف الرئيسي (MMenu.prg) . والسطور التالية توضح هذه الخطوات :

```
    * - - - USE original NewStock file, make all
    * - - - posted fields "True"
    CLOSE DATABASES
    USE NewStock
    REPLACE ALL Posted WITH .T.
ENDIF(record count > 0)

* - - - - - -  free up all work areas , and return to
* - - - - - -  Master menu.
CLOSE DATABASES
RETURN
```

١٧ - ٢ برنامج تصحيح ملف المبيعات (SalEdit.prg)

كما يلاحظ من عرض البرامج الفرعية المختلفة المكونة لبرنامج المخازن ، فقد تم تأجيل شرح البرنامج الخاص بتصحيح ملف المبيعات (SalEdit.prg) الى هذا

الفصل . والسبب فى ذلك أن التصحيح بعد عملية الترحيل يؤدى الى عدم مطابقة بيانات الملف الرئيسى (Master.dbf) للواقع . فمثلا لو فرضنا أن المستخدم باع عشرة طابعات ولكنه كتب فى حقل الكمية الخاص بهذا الصنف عشرين بدلاً من عشرة . فاذا قام بتعديل هذا الصنف فى ملف المبيعات بعد ترحيله الى الملف الرئيسى (Master) فان هذا التعديل لايظهر فى الملف الرئيسى . وبالتالى لايصبح الملف الرئيسى معبرا عن الكمية الصحيحة فى المخزن . ولذلك فان برنامج تصحيح ملف المبيعات يسمح للمستخدم باجراء أى تعديلات يريدها لملف المبيعات وفى نفس الوقت يقوم آليا بادخال هذه التعديلات على الملف الرئيسى . والخطوات المنطقية لتنفيذ هذه العملية تتلخص فى الآتى :

١- السماح للمستخدم بتعديل أى بيانات سجل معين .
٢- اذا قام المستخدم بتعديل بيان رقم جزء (Part number) فى ملف المبيعات . يقوم البرنامج باضافة الكمية المباعة من هذا الصنف على رقم الجزء السابق وطرحها من رقم الجزء الجديد فى الملف الرئيسى .
٣- اذا قام المستخدم بتعديل الكمية المباعة فى ملف المبيعات يتم حساب الفرق بين الكمية التى سبق ادخالها والكمية الجديدة . ثم يتم طرح هذه الكمية من الكمية الموجودة فى الملف الرئيسى .

وليست هناك حاجة لتعديل تاريخ آخر تحديث اذا قام المستخدم بتعديله فى ملف المبيعات لأنه فى جميع الأحوال يمثل تاريخ التحديث (Updating) وليس التعديل (Editing) .

وهذا البرنامج عند تشغيله يؤدى الى عرض الرسالة التالية على الشاشة .

```
Enter Invoice number to edit (0 if none) :
```

وعندما يكتب المستخدم رقم الفاتورة المطلوب التعديل فيها ، فان البرنامج يعرض كل الحركة الخاصة بهذه الفاتورة متضمنة بيانات الأصناف المختلفة . وفى هذه الحالة يقوم المستخدم باختيار أحد هذه الأصناف لتعديل بياناته . فتظهر شاشة خاصة بالتعديل متضمنة بيانات هذا الصنف لتعديلها أو مسحها . انظر الشكل(١٧ - ٢).

ويستطيع المستخدم تحريك المؤشر الى أى بيان خاص بهذا الصنف وتعديله . كما يمكنه كتابة (Y) أمام السؤال الذى يظهر أسفل الشاشة كالآتى :

```
Delete this record ? (Y/N)
```

وذلك عندما يريد الغاء هذا الصنف .

ويقوم البرنامج بعد ذلك باجراء التعديلات المطلوبة على الملف الرئيسى .

Edit Sales Transactions 02/20/90 10:50:30	
Invoice Number :	Date :
Part Number :	
Clerk :	Customer :
Quantity :	Selling Price:
Delete this record (Y/N)	

شكل (١٧ - ٢)

١٧ - ٢ - ١ الخطوات الأولية (PSEUDOCODE)

يتم كتابة الخطوات الأولية للبرنامج كالآتى :

١ - يتم انشاء متغيرات الذاكرة .
٢- يتم تكوين حلقة تكرارية لادخال رقم الفاتورة .
٣- يتم فتح ملف المبيعات (Sales.dbf) وملف الفهرس الخاص به (Sales.ndx) .
٤- يتم حساب عدد السجلات التى تشترك فى رقم الفاتورة الذى يدخله المستخدم .
٥- اذا لم يتم العثور على أى سجل بنفس رقم الفاتورة يتم تحذير المستخدم والسماح له بالمحاولة مرة ثانية مع رقم فاتورة آخر .
٦- اذا كانت هناك عدة سجلات تشترك فى رقم الفاتورة . يتم عرض بياناتها للمستخدم وسؤاله عن رقم السجل المطلوب تعديل بياناته . واذا كان هناك سجل واحد يتم الذهاب اليه .

٧- يتم الذهاب الى السجل المطلوب تعديل بياناته فى ملف المبيعات وتخزين القيم الموجودة فيه فى متغيرات ذاكرة .
٨- يتم عرض بيانات السجل من خلال شاشة تصحيح والسماح للمستخدم بتصحيح أى بيانات فى السجل .
٩- بعد انتهاء عملية التصحيح يتم ضبط بيانات الملف الرئيسى .
١٠- اذا كان السجل قد تم ترحيله (Posted) ثم قام المستخدم بمسحه يتم تجهيزه للمسح من الملف الرئيسى .
١١- اذا كان السجـــل قد تم ترحيله وقام المستخدم بتعديل رقم الجزء (Part number)، يتم الذهاب الى رقم الجزء القديم فى الملف الرئيسى (Master) واضافة الكمية التى سبق انقاصها منه . كما يتم الذهاب الى رقم الجزء الجديد فى الملف الرئيسى وطـــرح هذه الكمية منه .
١٢- اذا كان السجل قد تم ترحيله (Posted) وقام المستخدم بتعديل الكمية، يتم حساب الفرق بين الكمية السابقة والكمية الجديدة .
١٣- يتم الذهاب الى رقم الجزء فى الملف الرئيسى وطرح الفرق من الكمية الموجودة (Quantity) .
١٤- السماح للمستخدم بتكرار هذه العملية لأى جزء آخر .
١٥- اذا كانت هناك سجلات قد سبق تجهيزها للمسح يتم انهاء عملية المسح باستخدام الأمر (PACK) .
١٦- يتم الرجوع الى برنامج تشغيل ملف المبيعات (SMenu.prg) .

١٧ - ٢ - ٢ كتابة البرنامج

يتم كتابة سطور البرنامج كالآتى :

```
* * * * * * * * * * * * * * * * *SalEdit.prg
*     Edit the Sales File and update Master.
*     Called from Sales Menu , SMenu.prg.

*     Set up memory variables.
No_Dels = 0
Search = 1

* - - - - - Set up loop for invoice numbers .
DO WHILE Search # 0
    CLEAR
    @ 2,1  SAY "Edit Sales Transactions"
```

```
@ 2,60 SAY DTOC(T_Date) + "    " + TIME()
@ 3,0  SAY Uline
?
?
@ 15,5 SAY "Enter invoice number (0 if none) :" ;
  GET Search PICT "9999"
READ

* - - - - If user does not request exit, continue
* - - - - - with edit.
IF Search > 0
USE Sales INDEX Sales
* - - - Count records with that invoice number.
COUNT FOR invoice_no = Search TO Howmany
DO CASE
    * - - - If invoice not found , warn user.
    CASE Howmany = 0
        @ 24,1 SAY "No such invoice number"
        ? CHR(7)
    * - - - - -If invoice number found , proceed.
    CASE Howmany > 0
       * - - If several records have this invoice
       * - - number , dispaly them and get the
       * - - required record number
       IF Howmany > 1
          @ 5,1  CLEAR
          ? "RecNo  Part #  Qty  Price Clerk"
          ?? " Customer   Date"
          ?
          LIST FOR Invoice_no = Search Part_no , ;
          Qty, Price , Clerk, Customer , Date
          ?
          INPUT "Edit which record:" TO RecNo
          GOTO RecNo
       * - - - Otherwise, just go to the record.
       ELSE
          LOCATE FOR Invoice_no = Search
       ENDIF
```

```
* Store original field values to variables.
Old_Part = Part_no
Old_Qty  = Qty
* - - Display data on edit screen and allow
* - - edit.
@ 5,0 CLEAR
Deleted = "  "
@ 7,1  SAY "Invoice Number" GET Invoice_no
@ 7,44 SAY "Date" GET Date PICT "99/99/99"
@ 10,1 SAY "Part Number" GET Part_no;
  PICT "!!!!!"
@ 12,1 SAY "Clerk" GET Clerk
@ 12,40 SAY "Customer" GET Customer
@ 15,1 SAY "Qauntity" GET Qty
@ 15,20 SAY "Selling price" GET Price
@ 24,1 SAY "Delete this record? (Y/N)" ;
   GET Deleted PICT "!"
READ
* - - - After editing , adjust Master file.
DO CASE
   * - - - If transaction to be deleted ,
   * - - - delete it and add its quantity
   * - - - back to the Master file.
   CASE Posted .AND. Deleted = "Y"
     DELETE
     No_Dels = No_Dels + 1
     USE Master INDEX Master
     SEEK Old_Part
     IF FOUND()
         REPLACE Qty WITH Qty + Old_Qty
     ENDIF
     * - If Part number changed , add
     * - quantity to old part number,
     * - and subtract it from new part
     * - - - number.
   CASE Posted .AND. Part_no # Old_Part
     New_Qty = Qty
     New_Part = Part_no
```

```
                USE Master INDEX Master
                SEEK Old_Part
                IF FOUND()
                   REPLACE Qty WITH Qty + Old_Qty
                ENDIF
                * - -If quantity changed
                * - -adjust the Master file quantity
              CASE Posted .AND. Qty # Old_Qty
                Diff = Qty - Old_Qty
                USE Master INDEX Master
                SEEK Old_Part
                IF FOUND()
                   REPLACE Qty WITH Qty - Diff
                ENDIF
                SEEK New_Part
                IF FOUND()
                  REPLACE Qty WITH Qty - New_Qty
                ENDIF
           ENDCASE(adjustments after editing)
ENDCASE(Howmany > 0)
ENDIF(Search > 0)
ENDDO(while still editing)

* - - - -If there are records to be deleted , Pack the
* - - - -sales database file.
IF No_Dels > 0
   @ 5,0 CLEAR
   ? "Deleting unwanted records from the sales file.."
   USE Sales INDEX Sales
   PACK
ENDIF

* - - - - -  Return to Sales menu.
CLOSE DATABASES
RETURN
```

والبرنامج يبدأ بانشاء متغير الذاكرة (No_Dels) لتخزين عدد السجلات المطلوب مسحها ، وكذلك انشاء متغير الذاكرة (Search) لتخزين رقم الفاتورة التي يريد المستخدم تعديلها . ويتم ذلك من خلال السطور التالية :

```
   No_Dels = 0
   Search = 1
```

ثم يتم تكوين حلقة تكرارية لسؤال المستخدم عن رقم الفاتورة المطلوب تعديلها. وذلك من خلال السطور التالية :

```
DO WHILE Search # 0
CLEAR
@ 2,1  SAY "Edit Sales Transactions"
@ 2,60 SAY DTOC(T_Date) + "   " + TIME()
@ 3,0  SAY Uline
?
?
@ 15,5 SAY "Enter invoice number  (0 if none) :";
GET Search PICT "9999"
READ
```

ثم يتم فتح ملف المبيعات (Sales.dbf) والفهرس الخاص به للبحث عن رقم الفاتورة الذى أدخله المستخدم (فى حالة ادخال هذا الرقم) . ويتم حصر عدد السجلات التى تشترك فى هذا الرقم وتخزين هذا العدد فى المتغير (Howmany) . ويتم ذلك من خلال السطور التالية :

```
IF Search > 0
USE Sales INDEX Sales
* - - - - -  Count records with that invoice number.
COUNT FOR invoice_no = Search TO Howmany
```

اذا لم يكن هناك أى سجلات بنفس رقم الفاتورة يتم تحذير المستخدم وذلك من خلال السطور التالية :

```
DO CASE
    * - - - - -  If invoice not found , warn user.
    CASE Howmany = 0
```

```
1 SAY "No such invoice number"
(7)
```

اذا كان هناك عدة سجلات لها نفس رقم الفاتورة يقوم ال
بيانات هذه السجلات حتى يقوم المستخدم بتحديد رقم ال
تعديله . ثم يتم الذهاب الى هذا السجل لتعديله . ويتم ذل
السطور التالية :

```
ny > 0
 - If several records have this invoice
 - number , dispaly them and get the
 - required record number
wmany > 1
5,1  CLEAR
"RecNo  Part #  Qty  Price  Clerk"
 " Customer   Date"

ST FOR Invoice_no = Search Part_no , ;
y, Price , Clerk, Customer , Date

PUT "Edit which record:" TO RecNo
TO RecNo
```

اذا كان هناك سجل واحد برقم الفاتورة الذى تم ادخاله ي
هذا السجل . وذلك كالآتى :

```
R Invoice_no = Search
```

وفى الجزء التالى يتم تخزين رقم الجزء الأصلى والكمية ال
فى السجل الذى تم اختياره فى متغيرات ذاكرة قبل أن يقو
بتعديلها . وذلك حتى يمكن اختبار هذه الحقول بعد ذلك .
خلال السطور التالية :

```
= Part_no
= Qty
```

وفي الجزء التالى يتم انشاء متغير الذاكرة (Deleted) لتخزين رغبة المستخدم فى مسح السجل أو عدم مسحه . ثم يتم تصميم شاشة لتصحيح بيانات السجل . ويتم ذلك من خلال السطور التالية :

```
@ 5,0 CLEAR
Deleted = "  "
@ 7,1  SAY "Invoice Number" GET Invoice_no
@ 7,44 SAY "Date" GET Date PICT "99/99/99"
@ 10,1 SAY "Part Number" GET Part_no ;
  PICT "!!!!!"
@ 12,1 SAY "Clerk" GET Clerk
@ 12,40 SAY "Customer" GET Customer
@ 15,1 SAY "Qauntity" GET Qty
@ 15,20 SAY "Selling price" GET Price
@ 24,1 SAY "Delete this record? (Y/N)" GET ;
  Deleted PICT "!"
READ
```

وفي الجزء التالى من البرنامج يتم اختبار هذا السجل بعد تعديله بواسطة المستخدم . فاذا كان السجل قد تم ترحيله (Posted) وطلب المستخدم مسحه ("Y" = Deleted) يتم تجهيزه للمسح بواسطة الأمر (DELETE) ثم يتم زيادة عدد السجلات المطلوب مسحها (No_Dels) بواحد .

ثم يقوم البرنامج بعد ذلك بفتح الملف الرئيسى (Master) وملف الفهرس الخاص به للبحث عن هذا السجل المطلوب مسحه واضافة الكمية التى عدلها المستخدم مرة ثانية . ويتم ذلك من خلال السطور التالية :

```
DO CASE
    * - - - - - -  If transaction to be deleted ,
    * - - - - - -  delete it and add its quantity
    * - - - - - -  back to the Master file.
    CASE Posted .AND. Deleted = "Y"
    DELETE
    No_Dels = No_Dels + 1
    USE Master INDEX Master
```

```
SEEK Old_Part
IF FOUND()
    REPLACE Qty WITH Qty + Old_Qty
ENDIF
```

وفي الجزء التالى يتم اختبار حالة ثانية فى السجل ، وهي عندما يقوم المستخدم بتعديل رقم الجزء . وذلك عندما يكتشف مثلا أنه أدخل رقم أحد الأجزاء خطا عند كتابة فاتورة البيع . وفي هذه الحالة تكون كمية الصنف (Quantity) الخاصة بهذا الجزء قد نقصت في الملف الرئيسي (Master) في حين تظل كمية الصنف الذى كان يجب ادخال رقمه كما هي . ولذلك يقوم البرنامج باضافة الكمية التى سبق خصمها من الصنف فى الملف الرئيسى الى الجزء القديم (Old_Part) وخصمها من كمية الصنف فى الجزء الجديد . ويتم ذلك من خلال السطور التالية :

```
CASE Posted .AND. Part_no # Old_Part
    New_Qty = Qty
    New_Part = Part_no
    USE Master INDEX Master
    SEEK Old_Part
    IF FOUND()
        REPLACE Qty WITH Qty + Old_Qty
    ENDIF
```

والجزء التالى يعالج الحالة الثالثة وهي عندما يعدل المستخدم الكمية (Quantity) فقط دون تعديل رقم الجزء . في هذه الحالة يقوم البرنامج بتعديل كمية هذا الجزء في الملف الرئيسي (Master) ، وذلك بخصم الكمية القديمة من الكمية الجديدة التى عدلها المستخدم وتخزين هذا الفرق في المتغير (DIFF) . ثم يتم طرح هذا الفرق (DIFF) من الكمية الحالية الموجودة فى الملف الرئيسي ويتم ذلك من خلال السطور التالية :

```
CASE Posted .AND. Qty # Old_Qty
    Diff = Qty - Old_Qty
    USE Master INDEX Master
    SEEK Old_Part
    IF FOUND()
        REPLACE Qty WITH Qty - Diff
    ENDIF
ENDCASE(adjustments after editing)
```

```
ENDCASE(Howmany > 0)
ENDIF(Search > 0)
ENDDO(while still editing)
```

وفى الجزء التالى يتم اختبار المتغير (No_Dels) لمعرفة ما اذا كان هناك سجلات مطلوب مسحها أم لا . فاذا كانت هناك سجلات مطلوب مسحها يتم استخدام الأمر (PACK) في مسحها نهائيا من ملف المبيعات (Sales.dbf) . ويتم ذلك من خلال السطور التالية :

```
IF NO_Dels > 0
   @ 5,0 CLEAR
   ? "Deleting unwanted records from the sales file.."
   USE Sales INDEX Sales
   PACK
ENDIF
```

وفى نهاية البرنامج يتم اغلاق الملفات والعودة الى برنامج المبيعات (SMenu.prg) .

١٧ - ٣ برنامج تصحيح ملف الاضافة (NewEd.prg)

يتم تشغيل هذا البرنامج عندما يختار المستخدم الاختيار رقم (3) فى قائمة برنامج الاضافة . انظر الشكل (١٧ - ٣) .

```
New stock system menu        02/20/90    08:30:45

      1. Record new items
      2. Print new stock data
      3. Edit new stock data
      4. Return to main menu

   Enter choice   ( 1 - 4 )
```

شكل (١٧ - ٣)

وهذا البرنامج مثل البرنامج السابق يسمح للمستخدم بتعديل سجلات اضافة الأصناف التي سبق ادخالها ، بحيث يتم تعديل الملف الرئيسي (Master) بناء على هذه التعديلات اذا كانت السجلات قد سبق ترحيلها (Posted) . وهو لايختلف في تركيبه عن البرنامج السابق ولكنه يختلف عنه في الحاجة الى تعديل حقل سعر الصنف (Cost) وحقل (On_Order) بالاضافة الى حقل الكمية (Qty). كما أن البرنامج يستخدم رقم الجزء في البحث عن الصنف المطلوب في ملف الاضافة (NewStock.dbf) وليس رقم الفاتورة مثل ملف المبيعات . ويتكون البرنامج من السطور التالية :

```
* * * * * * * * * * * * * * * * *NewEdit.prg
*    Edit the NewStock File and update Master.
*    Called from NMenu.prg.
*    Set up memory variables.
No_Dels = 0
Search = "1"

* - - - - - Get part number for data to edit.
DO WHILE Search # "0"
    Use NewStock INDEX NewStock
    CLEAR
    @ 2,1  SAY "Edit NewStock Transactions"
    @ 2,60 SAY DTOC(T_Date) + "   " + TIME()
    @ 3,0  SAY Uline
    ?
    ACCEPT "Enter part number to edit (0 if none)" ;
      TO Search
    Search = UPPER(Search)
    * - - - - If user did not request exit, continue.
    IF Search # "0"
       * - - Count the records with that part number
       SEEK Search
       COUNT WHILE Part_no = Search TO Howmany
       DO CASE
          * - - - If part not found , warn user.
          CASE Howmany = 0
          ? "  No such part number"
          ? CHR(7)
```

```
* - - - If part number found , proceed.
CASE Howmany > 0
  * - -  If several records have that part
  * - -  number , dispaly them and get the
  * - -  required record
  IF Howmany > 1
    CLEAR
    SEEK Search
    LIST WHILE Part_no = Search Part_no , ;
      Qty, Cost, Date, Vendor
    ?
    INPUT "Edit which record(enter ;
     record number)?" TO RecNo
    GOTO RecNo
  ELSE
    SEEK Search
    RecNo = RECNO()
  ENDIF(Howmany > 1)

  * Store original field values to variables
  Old_Part = Part_no
  Old_Qty = Qty
  Old_Cost = Cost
  *- - Display edit screen and allow edit.
  CLEAR
  Deleted = "  "
  @ 1,1  SAY "Edit NewStock Transaction"
  @ 3,1  SAY "Part Number" GET Part_no ;
    PICT "!!!!!"
  @ 5,1  SAY "Quantity" GET Qty
  @ 5,20 SAY "Purchase Price" GET Cost
  @ 7,1  SAY "Date" GET Date PICT "99/99/99"
  @ 7,15 SAY "Vendor" GET Vendor
  @ 9,2  SAY "Delete this record? (Y/N)" ;
    GET Deleted PICT "!"
  READ

  * - After editing , adjust Master file.
```

```
* - -  First handle change in cost.
IF Cost # Old_Cost
    New_Part = Part_no
    New_Cost = Cost
    USE Mater INDEX Master
    SEEK New_Part
    IF FOUND()
      REPLACE Cost WITH New_Cost
      USE NewStock INDEX NewStock
      GOTO RecNo
    ENDIF
ENDIF (Cost # Old_Cost)
DO CASE
   * - - -if new stock transaction to
   * - - -be deleted, subtract its
   * - - -quantity from the Master
   * - - -file and add it to the
   * - - -On_order field.
   CASE Posted .AND. Deleted = "Y"
      DELETE
      No_Dels = No_Dels + 1
      USE Master INDEX Master
      SEEK Old_Part
      IF FOUND()
         REPLACE Qty WITH Qty + ;
         Old_Qty
         REPLACE On_order WITH ;
           On_order + Old_Qty
      ENDIF
      * - - If Part number changed,
      * - - subtract quantity from the old
      * - - part number, and add to the
      * - - new . Do the opposit for the
      * - - On_order field.
      CASE Posted .AND. Part_no # Old_Part
         New_Qty = Qty
         New_Part = Part_no
         USE Master INDEX Master
```

```
                       SEEK Old_Part
                       IF FOUND()
                          REPLACE Qty WITH Qty - Old_Qty
                          REPLACE On_order WITH On_order;
                           + Old_Qty
                        ENDIF
                        SEEK New_Part
                        IF FOUND()
                          REPLACE Qty WITH Qty + New_Qty
                          REPLACE On_order WITH On_order;
                            - New_Qty
                        ENDIF
                        * - - If user just changed the
                        * - - quantity , adjust the
                        * - - Master file quantity.
                        CASE Posted .AND. Qty # Old_Qty
                           Diff = Qty - Old_Qty
                           USE Master INDEX Master
                           SEEK Old_Part
                           IF FOUND()
                              REPLACE Qty WITH Qty + Diff
                              REPLACE On_order WITH ;
                                On_order - Diff
                           ENDIF
            ENDCASE
        ENDCASE
    ENDIF(Search # 0).
ENDDO(while Search # 0)

* - - -   If records have been deleted , Pack the New
* - - - - Stock data file.
IF NO_Dels > 0
    CLEAR
    ? "Deleting unwanted records from NewStock file"
    USE NewStock INDEX NewStock
    PACK
ENDIF
```

```
* - - - - -  Done , Return to NewStock Menu.
CLOSE DATABASES
RETURN
```

والبرنامج يبدأ بانشاء متغيرات الذاكرة مثل البرنامج السابق تماما . ولكن يلاحظ أن المتغير (Search) يتم انشاؤه كمتغير حرفي ، وذلك لأن البحث سوف يتم عن رقم الجزء وليس رقم الفاتورة مثل البرنامج السابق . ورقم الجزء يمكن أن يحتوى على أرقام وحروف . ولذلك تستخدم الدالة (UPPER) فى تحويل رقم الجزء الذى يدخله المستخدم الى حروف كبيرة حتى يماثل الفهرس . والسطور التالية توضح هذا الجزء

```
No_Dels = 0
Search = "1"

* - - - - - Get part number for data to edit.
DO WHILE Search # "0"
    Use NewStock INDEX NewStock
    CLEAR
    @ 2,1  SAY "Edit NewStock Transactions"
    @ 2,60 SAY DTOC(T_Date) + "   " + TIME()
    @ 3,0  SAY Uline
    ?
    ACCEPT "Enter part number to edit(0 if none)" TO;
    Search
    Search = UPPER(Search)
```

ثم يتم البحث عن رقم الجزء الذى أدخله المستخدم وحصر عدد السجلات التى لها هذا الرقم وتخزين هذا العدد فى المتغير (Howmany) . ويتم ذلك من خلال السطور التالية :

```
IF Search # "0"
* - - - - -  Count the records with that part number
SEEK Search
COUNT WHILE Part_no = Search TO Howmany
```

واذا لم يتم العثور على هذا الرقم يتم تحذير المستخدم وذلك كالآتى :

```
DO CASE
    * - - - - -  If part not found , warn user.
    CASE Howmany = 0
    ? "  No such part number"
    ? CHR(7)
```

واذا كان هناك عدة سجلات لها نفس رقم الجزء المطلوب ، يتم عرض بيانات هذه السجلات وسؤال المستخدم عن رقم السجل المطلوب . ويتم ذلك من خلال السطور التالية :

```
IF Howmany > 1
    CLEAR
    SEEK Search
    LIST WHILE Part_no = Search Part_no , ;
    Qty, Cost, Date, Vendor
    ?
    INPUT "Edit which record(enter record ;
    number) ?" TO RecNo
    GOTO RecNo
```

واذا كان هناك سجل واحد يحتوى على رقم الجزء المطلوب يتم الذهاب الى هذا السجل وتخزين رقم السجل فى متغير الذاكرة (RecNo) ويتم ذلك من خلال السطور التالية :

```
ELSE
    SEEK Search
    RecNo = RECNO()
ENDIF(Howmany > 1)
```

وفى الجزء التالى يتم تخزين محتويات بعض حقول السجل المطلوب تعديله فى متغيرات ذاكرة حتى يمكن استخدامها بعد ذلك فى تعديل الملف الرئيسى . ويتم ذلك من خلال السطور التالية :

```
old_part = part_no
old_Qty  = Qty
old_Cost = Cost
```

وفى الجزء التالى يقوم البرنامج بعرض بيانات سجل الاضافة للمستخدم حتى يمكنه تعديل الحقول المطلوبة أو مسح السجل من ملف الاضافة (NewStock.dbf) حسب الحاجة . ويتم ذلك من خلال السطور التالية :

```
CLEAR
Deleted = "  "
@ 1,1  SAY "Edit NewStock Transaction"
@ 3,1  SAY "Part Number" GET Part_no ;
  PICT "!!!!!"
@ 5,1  SAY "Quantity" GET Qty
@ 5,20 SAY "Purchase Price" GET Cost
@ 7,1  SAY "Date" GET Date PICT "99/99/99"
@ 7,15 SAY "Vendor" GET Vendor
@ 9,2  SAY "Delete this record? (Y/N)" GET ;
  Deleted PICT "!"
READ
```

وفى الجـزء التالى يقوم البرنامج بضبط البيانات فى الملف الرئيسى (Master.dbf) بناء على البيانات التى تم تعديلها فى ملف الاضافة (NewStock.dbf) . فاذا كان المستخدم عدل سعر الجزء (Cost) يتم تعديله فى الملف الرئيسى . وذلك من خلال السطور التالية :

```
IF Cost # Old_Cost
    New_Part = Part_no
    New_Cost = Cost
    USE Mater INDEX Master
    SEEK New_Part
    IF FOUND()
        REPLACE Cost WITH New_Cost
        USE NewStock INDEX NewStock
        GOTO RecNo
    ENDIF
ENDIF(Cost # Old_Cost)
```

وفى الجزء التالى يتم اختبار حالة أخرى ، عندما يطلب المستخدم مسح هذا السجل (Deleted = "Y") ، فان البرنامج يقوم بتجهيزه للمسح باستخدام الأمر (DELETE) ، ثم يقوم بفتح الملف الرئيسى والفهرس الخاص به . ويتم البحث عن

هذا السجل فاذا كان موجودا في الملف الرئيسي يتم خصم الكمية التي سبق اضافتها من الملف الرئيسي ، كما يتم اضافة هذه الكمية الى الكمية تحت الطلب (On_order) . وذلك من خلال السطور التالية :

```
DO CASE
CASE Posted .AND. Deleted = "Y"
    DELETE
    No_Dels = No_Dels + 1
    USE Master INDEX Master
    SEEK Old_Part
    IF FOUND()
        REPLACE Qty WITH Qty - Old_Qty
        REPLACE On_order WITH ;
        On_order + Old_Qty
    ENDIF
```

والجزء التالى يعالج حالة أخرى عندما يقوم المستخدم بتعديل رقم الجزء الذى سبق الاضافة عليه . وفي هذه الحالة يقوم البرنامج بخصم الكمية المضافة من الجزء السابق واضافتها الى الجزء الجديد . كما يتم اضافة هذه الكمية الى الكمية تحت الطلب . ويتم ذلك من خلال السطور التالية :

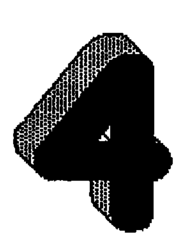
```
CASE Posted .AND. Part_no # Old_Part
    New_Qty = Qty
    New_Part = Part_no
    USE Master INDEX Master
    SEEK Old_Part
    IF FOUND()
        REPLACE Qty WITH Qty - Old_Qty
        REPLACE On_order WITH On_order;
          + Old_Qty
    ENDIF
    SEEK New_Part
    IF FOUND()
        REPLACE Qty WITH Qty - New_Qty
        REPLACE On_order WITH On_order;
          - New_Qty
    ENDIF
```

والجزء التالى يعالج الحالة الأخيرة عندما يقوم المستخدم بتعديل الكمية التى سبق اضافتها . في هذه الحالة يتم ضبط الكمية فى الملف الرئيسى بناء على الفرق بين الكمية الأولى والكمية الجديدة بعد التعديل . والسطور التالية توضح هذه العملية :

```
CASE Posted .AND. Qty # Old_Qty
    Diff = Qty - Old_Qty
    USE Master INDEX Master
    SEEK Old_Part
    IF FOUND()
        REPLACE Qty WITH Qty + Diff
        REPLACE On_order WITH ;
         On_order - Diff
    ENDIF
ENDCASE
ENDCASE
ENDIF(Search # 0)
ENDDO(while Search # 0)
```

وفى الجزء التالى يتم مسح السجلات التى سبق تجهيزها للمسح . ويتم ذلك باستخدام الأمر (PACK) . ثم يتم اغلاق الملفات والعودة الى برنامج الاضافة (NMenu.prg) . والسطور التالية توضح هذه العملية :

```
IF NO_Dels > 0
  CLEAR
  ?"Deleting unwanted records from the NewStock file"
  USE NewStock INDEX NewStock
  PACK
ENDIF
CLOSE DATABASES
RETURN
```



الجزء الرابع

نظام حسابات العملاء

* تصميم النظام
* ملفات الخطوات
* برنامج القائمة الرئيسية والادخال والتعديل
* تقارير برنامج حسابات العملاء
* التحديث الشهرى للنظام



الفصل الثامن عشر

تصميم النظام



يتم فى هذا الجزء تصميم نظام حسابات العملاء (Accounts Receivable) الذى يشتمل على طباعة الفواتير آليا . ويتم فى هذا النظام تخزين اسم العميل (Customer) وعنوانه فى ملف قاعدة بيانات منفصل بالاضافة الى الموازنة الحالية (Current balance) والموازنات السابقة (Aged balances) . كما يتم تخزين بيانات السداد (Payments) فى ملف قاعدة بيانات منفصل . كما يتم تخزين بيانات الصرف (Charges) فى ملف قاعدة بيانات اخر ويقوم البرنامج بالربط بين هذه الملفات الثلاثة .

ودراسة هذا النظام تتيح لمخطط البرامج معرفة وسائل جديدة للتعامل مع الملفات المرتبطة (Relational Databases) . كما يتم من خلالها دراسة استخدام ملفات الخطوات (Procedure Files) فى تسهيل كتابة البرامج وزيادة كفاءتها .

١٨ - ١ تعريف المشكلة

الهدف من نظام حسابات العملاء هو انشاء قاعدة بيانات يتم من خلالها متابعة بيانات العملاء ، وتسجيل الموقف الشهرى (Monthly) لحركة السداد (Payments Transactions)وحركة الصرف (Charges Transactions) لكل حساب (Account) . ويقوم النظام باصدار فواتير (Invoices) لكل حساب توضح الموازنة الابتدائية (Starting Balance) للشهر ، وكل حركة سداد أو صرف خلال هذا الشهر ، ثم الموازنة الحالية (Current Balance) . كما يوفر النظام تقارير مختصرة (Summary Reports)للنشاط الشهرى لكل عميل . كما يوفر أيضا تقارير للموازنة خلال ٣٠ يوما ، ٦٠ يوما ، ٩٠ يوما وأكثر من ٩٠ يوما . كما يتيح النظام أيضا تخزين نسخة من فاتورة العميل حتى يستطيع العميل مراجعة أى بيانات سابقة .

ويقوم البرنامج كالعادة باستخدام القوائم الواضحة فى تسهيل تعامل المستخدم مع النظام .

١٨ - ٢ تحديد هيكل قاعدة البيانات

كما سبق الايضاح فان قاعدة البيانات تتكون من ثلاثة ملفات . أولها ملف بيانات العميل (Customer) الذى يحتوى على كل البيانات الأساسية لهذا العميل . والثانى ملف حركة الصرف (Charges) الذى يتضمن بيانات فواتير الصرف والأجزاء المصروفة وأسعارها . والثالث ملف حركة السداد (Payments) الذى يتضمن

بيانات الشيكات وموقف تسديد الفواتير .

١٨ - ٢ - ١ ملف بيانات العميل (Custamer.dbf)

يتكون ملف بيانات العميل من الحقول الموضحة بالشكل (١٨ - ١)

Field	Field Name	TYPE	Width	Dec
1	ACCOUNT_NO	Numeric	4	0
2	CUST_NAME	Character	25	
3	ADDRESS	Character	25	
4	PHONE	Character	13	
5	LAST_UPDAT	Date	8	
6	START_BAL	Numeric	8	2
7	CHG_CURR	Numeric	8	2
8	PAY_CURR	Numeric	8	2
9	BAL_30	Numeric	8	2
10	BAL_60	Numeric	8	2
11	BAL_90	Numeric	8	2
12	BAL_90, PLUS	Numeric	8	2
13	TERMS	Character	20	

شكل (١٨ - ١)

والحقل رقم (١) يمثل رقم الحساب وهو حقل عددى . والحقل رقم (٢) يمثل اسم العميل وهو حقل حرفى يتكون من ٢٥ حرفا . والحقل رقم (٣) يمثل العنوان وهو حقل حرفى يتكون من ٢٥ حرفا . والحقل رقم (٤) هو حقل رقم التليفون وهو حقل حرفى يتكون من (١٣) حرفا . والحقل رقم (٥) هو حقل تاريخ آخر تعديل . والحقل رقم (٦) هو حقل الموازنة الابتدائية وهو حقل عددى مكون من ثمانية أرقام ورقمين عشريين . والحقل رقم (٧) هو حقل حركة الصرف الحالية (Current Charges) . والحقل رقم (٨) هو حقل حركة السداد

الحالية (Current Payment) . والحقل رقم (٩) هو حقل الموازنة خلال شهر . والحقل رقم (١٠) هو حقل الموازنة خلال شهرين . والحقل رقم (١١) هو حقل الموازنة خلال ثلاثة شهور . والحقل رقم (١٢) هو حقل الموازنة خلال أربعة أشهر . والحقل رقم (١٣) هو حقل الدين (Credit).

ويستخدم حقل رقم الحساب (Account_No) فى ربط الملفات الثلاثة . ولذلك فمن الطبيعى أن تتم فهرسة الملفات الثلاثة بناء على هذا الحقل . ولفهرسة ملف العميل (Cnstomer. dbf) يتم كتابة السطر التالى من مشيرة النقطة (Dot Prompt) .

```
INDEX ON Account_No TO Cust_No
```

ولكن قد يحتاج المستخدم فى بعض الأحيان الى البحث عن عميل معين باسمه ، لذلك يمكن انشاء فهرس آخر بناء على حقل اسم العميل (Cust_name) . ويمكن استخدام الدالة (UPPER) فى تحويل اسم العميل فى الفهرس الى حروف كبيرة (Uppercase) لتسهيل البحث عن اسم العميل سواء أدخل المستخدم الاسم بحروف صغيرة أو حروف كبيرة ، ولانشاء هذا الفهرس يتم كتابة السطر التالى من مشيرة النقطة (Dot Prompt) .

```
INDEX ON UPPER(Cust_name) TO Cust_Name
```

١٨ - ٢ - ٢ ملف حركة الصرف (Charges.dbf)

يتكون ملف حركة الصرف (Charges.dbf) من الحقول الموضحة فى الشكل (١٨ - ٢) . ويلاحظ أن حقل رقم الحساب (Account_no) موجود فى الملف بنفس الاسم ونفس النوع (Type) ونفس العرض (Width) ، وذلك لأنه الحقل الذى سوف يستخدم كمفتاح (Key Field) لربط الملفات الثلاثة .

ويتم انشاء فهرس لهذا الملف بناء على حقل رقم الحساب ، وذلك بكتابة السطر التالى من مشيرة النقطة (Dot Prompt).

```
INDEX ON Account_No TO ChrgNo
```

Field	Field Name	TYPE	Width	Dec
1	ACCOUNT_NO	Numeric	4	0
2	INVOICE_NO	Numeric	6	0
3	PART_NO	Character	5	
4	QTY	Numeric	7	2
5	UNIT_PRICE	Numeric	9	2
6	AMOUNT	Numeric	9	2
7	DATE	Date	8	
8	DESCRIPT	Character	20	
9	BILLED	Logical	1	

شكل (١٨ - ٢)

والحقل رقم (١) يمثل رقم الحساب ، والحقل رقم (٢) يمثل رقم الفاتورة ، والحقل رقم (٣) يمثل حقل رقم الجزء ، والحقل رقم (٤) يمثل حقل الكمية ، والحقل رقم (٥) يمثل حقل سعر الوحدة . والحقل رقم (٦) يمثل حقل الكمية الكلية المصروفة ، والحقل رقم (٧) يمثل حقل تاريخ الشراء، والحقل رقم (٨) يمثل حقل وصف عملية الشراء ، والحقل رقم (٩) هو حقل منطقى يوضح اذا كان الحساب قد تم تسديده أم لا .

١٨ - ٢ - ٣ ملف السداد (Payments.dbf)

أما ملف السداد (Payments.dbf) فيتم تكوينه من الحقول الموضحة فى الشكل (١٨ - ٣) .

والحقل رقم (١) يمثل رقم الحساب ، والحقل رقم (٢) يمثل رقم الشيك ، والحقل رقم (٣) يمثل حقل الكمية الكلية المدفوعة ، والحقل رقم (٤) يمثل حقل تاريخ الشراء ، والحقل رقم (٥) يمثل حقل وصف العملية ، والحقل رقم (٦) هو حقل منطقى يوضع اذا كان الدفع قد تم تسجيله أم لا .

Field	Field Name	TYPE	Width	Dec
1	ACCOUNT_NO	Numeric	4	0
2	CHECK_NO	Character	5	
3	AMOUNT	Numeric	9	2
4	DATE	Date	8	
5	DESCRIPT	Character	30	
6	POSTED	Logical	1	

شكل (١٨ - ٢)

ويتم انشاء ملف الفهرس لهذا الملف بكتابة السطر التالى من مشيرة النقطة (Dot Prompt) كالآتى :

```
INDEX ON Cust_no TO PayNo
```

١٨ - ٣ حفظ البيانات التاريخية

يقوم نظام حسابات العملاء كما سبق الايضاح على المتابعة الشهرية ، أى تسجيل حركة الصرف والتوريد خلال الشهر . وفى نهاية الشهر يتم تجميع الفواتير وارسالها الى العملاء .

ولكن ماذا لو أراد المستخدم الرجوع الى بيانات سابقة عن حسابات تم تسديدها؟

هناك ثلاثة احتمالات للتعامل مع البيانات القديمة :

١- مسح كل حركة للصرف أو السداد بمجرد تسديدها . وتمتاز هذه الطريقة بتوفير مساحة تخزينية على القرص ؛ ولكن يعيبها عدم القدرة على الرجوع الى أى بيانات سابقة عن أى حركة تم تسديدها .

٢- ترك كل حركة صرف أو سداد مخزنة في الملف حتى بعد تسديدها، مع تمييز الحركة التي تم تسديدها . وفائدة هذه الطريقة أنها تحتفظ بالبيانات السابقة عن أي حركة مهما كان تاريخها . ولكن يعيبها أنها سوف تؤدي الى تضخم ملفي الصرف (Charges) والسداد (Payment) الى حجم لانهائي .

٣ - نقل كل حركة صرف أو سداد بمجرد تسديدها من ملفي الصرف والسداد الى ملف تاريخي (Historical file) . وفائدة هذه الطريقة أنها تؤدي الى الاحتفاظ بملفي الصرف والتوريد صغيرين نسبيا مع الاحتفاظ بجميع البيانات السابقة عن أي حركة سبق تسديدها .

ومن خلال نظام حسابات العملاء الجاري شرحه سيتم استخدام الاحتمال الثالث ، وهو نقل جميع السجلات التي تم تسديدها من ملفي الصرف والسداد الى ملف تاريخي (Historical File) . ويمكن تخزين هذا الملف التاريخي في قرص أو أقراص منفصلة . ولانشاء ملف تاريخي لملف الصرف (Charges.dbf) يتم نسخ هيكل الملف (Structure) الى ملف جديد يتم تسميته (BillHist.dbf) مثلا . وذلك بكتابة السطرين التاليين من مشيرة النقطة (Dot Prompt) .

```
USE Charges
COPY STRUCTURE TO BillHist
```

ولانشاء ملف تاريخي مماثل بالنسبة لملف السداد (Payments) يتم كتابة السطرين التاليين من مشيرة النقطة .

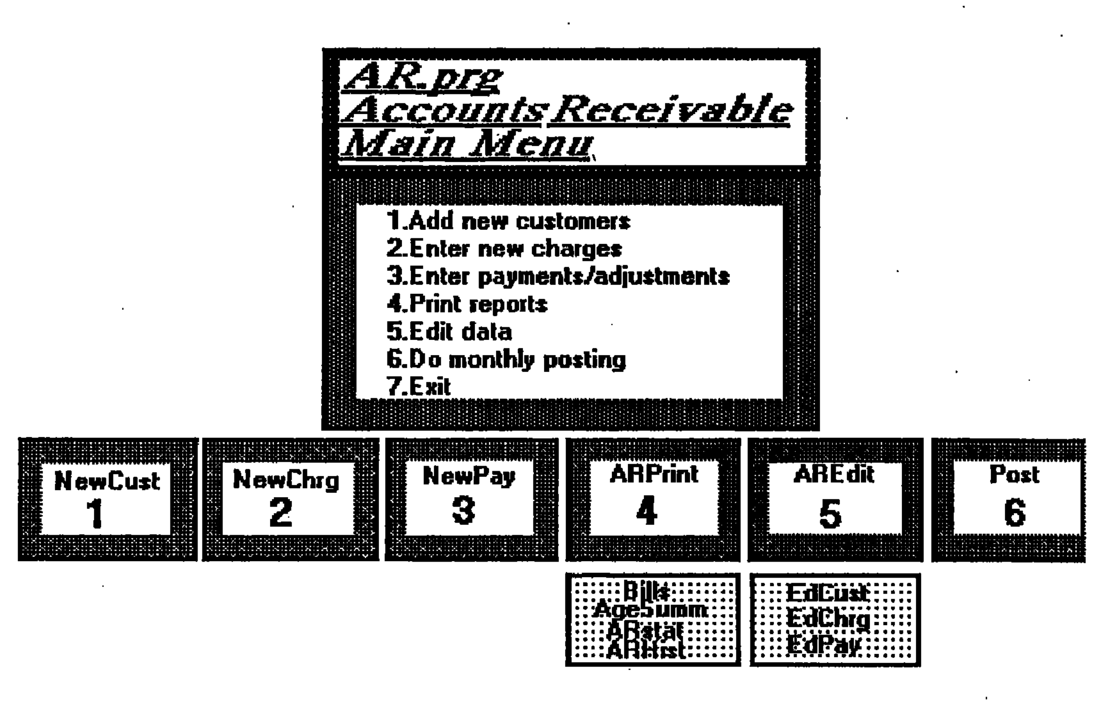
```
USE Payments
COPY STRUCTURE TO PayHist
```

وفي الأجزاء التالية سيتم ايضاح كيفية نقل السجلات التي تصبح غير مطلوبة الى الملفات التاريخية من خلال برنامج حسابات العملاء .

١٨ - ٤ تركيب البرنامج

يتكون برنامج حسابات العملاء من عدة برامج منفصلة يقوم أحدها باضافة عملاء جدد ، والثاني بطباعة التقارير والفواتير ، والثالث بتعديل بيانات أحد العملاء ، والرابع بالترحيل الشهري و ... الخ .

والشكل الهرمي (Hierarchical) للبرنامج يتضح من الشكل (١٨ - ٤) . ويلاحظ من الشكل أن البرنامج أكثر تعقيدا من برنامج المخازن حيث أنه يشمل عدة ملفات . ولذلك سيتم في الأجزاء التالية استخدام وسيلة جديدة لزيادة سرعة تنفيذ البرنامج وتقليل عدد الملفات المستخدمة . وهذه الوسيلة هى استخدام ملف الخطوات (Procedure File) وسيتم شرحها فى الباب التالى .



شكل (١٨ - ٤)



الفصل التاسع عشر

ملفات الخطوات



في هذا الفصل سيتم شرح احدى الوسائل المتقدمة لكتابة البرامج بواسطة (DBase III+) أو برامج عائلة (DBase) الأخرى ، وهي استخدام ملفات الخطوات (Procedure Files) والمعاملات (Parameters) . وسوف يتم استخدام هذه الوسيلة في برنامج حسابات العملاء (A/R) .

وملف الخطوات هو ملف يحتوى على برامج صغيرة (Routines) ، ويمكن استخدام كل برنامج في عدة أماكن من البرنامج الرئيسي باستدعائها عند الحاجة اليها أي أنه يعتبر مكتبة برامج (Library) . ويمكن انشاء عدة ملفات خطوات . ولكن لايمكن فتح أكثر من ملف في نفس الوقت . وهي تؤدي الى تسهيل كتابة البرامج وسرعة تنفيذها كما أنها تتغلب على مشكلة زيادة عدد الملفات المفتوحة التي تظهر في البرامج الكبيرة . وذلك لأن ملف الخطوات يعتبر ملفا واحدا رغم أنه يحتوى على عدة برامج .

ارجع الى الأمر (SET PROCEDURE TO) في الكتاب السابق .

١٩ - ١ استخدام ملف الخطوات في برنامج حسابات العملاء

يستخدم مع برنامج حسابات العملاء ملف خطوات واحد يحتوى على ثلاثة برامج . وهذا الملف عند فتحه باستخدام الأمر (SET PROCEDURE TO) يتم تحميله في الذاكرة المؤقتة (RAM) . وهذا يؤدى الى سرعة تشغيل البرامج داخله حيث لاتصبح هناك حاجة دائما الى نقل هذه البرامج من القرص الى الذاكرة .

وكل برنامج (Procedure) موجود داخل الملف يمكن تحويله الى برنامج عام (General) يمكن استخدامه في أي برامج أخرى . ويتم ذلك باستخدام الأمر (PARAMETERS) . وهذا الأمر يؤدي الى ادخال معاملات معينة الى البرنامج ليناسب تطبيقا محددا . ويمكن استخدام معاملات أخرى لتنفيذ البرنامج في تطبيقات أخرى متعددة . وهذا سيتم ايضاحه عند دراسة البرامج الثلاثة (Procedures) المستخدمة مع برنامج حسابات العملاء .

١٩ - ٢ برنامج العنوان (Title)

يستخدم هذا البرنامج لكتابة أي عنوان وكتابة تاريخ اليوم الحالي مع رسم خط عرضي بعرض الشاشة . ويمكن استخدام هذا البرنامج مع أي شاشة يتم تصميمها للمستخدم .

ويبدأ هذا البرنامج بالأمر (PROCEDURE) يليه اسم البرنامج (Title) ، وذلك

فى السطر الأول . وفى السطر الثاني يتم كتابة الأمر (PARAMETERS) يليه اسم المعامل أو المعاملات التي سوف يتم استخدامها عند استدعاء البرنامج . والسطر التالى يقوم بمسح الشاشة ، والسطر التالى يقوم بكتابة العنوان (Title) ، والسطر التالى يقوم بكتابة تاريخ اليوم الحالى والوقت الحالي ، والسطر التالى يقوم برسم خط عرضى بعرض الشاشة (Uline) . مع ملاحظة أن المتغير(Uline) يكون قد سبق تعريفه في البرنامج الرئيسى كما تم فى برنامج المخازن . وسطور البرنامج التى سبق شرحها تكون كالآتى :

```
PROCEDURE Title
PARAMETERS Title
    * - - - - - -  Display screen title
    CLEAR
    @ 2,1  SAY Title
    @ 2,60 SAY DTOC(DATE() + "  " + TIME()
    @ 3,0  SAY Uline
    ?
    ?
 RETURN
```

والشيء المثير فى هذه العملية أنه فى أى نقطة من البرنامج الرئيسى عندما يراد عرض شاشة للمستخدم متضمنة العنوان وتاريخ اليوم الحالى والوقت والخط العرضى ، فانه يكفي كتابة الأمر (DO) يليه اسم هذا البرنامج ثم العنوان المراد كتابته . وذلك كالآتى مثلا :

```
Do Title WITH "Accounts Receivable Main Menu"
```

وكلمة (WITH) تنبه برنامج (DBase III+) أن البيانات التالية لها هى المعاملات التي سبق تعريفها فى البرنامج الفرعى (Procedure) . ولذلك فعند تنفيذ هذا البرنامج الفرعي ، وعند وصول البرنامج الى الأمر (PARAMETERS) فانه يستبدل العنوان (Title) بالعنوان الذى تم ادخاله وهو ("Account Receivable Main Menu") . لذلك فعند تنفيذ البرنامج يظهر الآتى على الشاشة .

```
Accounts Receivable Main Menu    02/20/90    12:50:30
-----------------------------------------------------
```

١٩ - ٣ برنامج رسائل الأخطاء

من البرامج التى تستخدم أيضا فى ملفات الخطوات (Procedure Files) برنامج رسائل الأخطاء . وهذا البرنامج يستخدم فى عرض رسائل الأخطاء على الشاشة لتوضح للمستخدم الخطأ الذى وقع فيه والعمل الذى يجب تنفيذه للتغلب على هذا الخطأ . وهذا البرنامج يتكون من السطور التالية :

```
PROCEDURE Error
PARAMETERS Message
    * - - - - - -  Display error message.
    @ 20,0 CLEAR
    @ 20,3 SAY Message
    ? CHR(7)
    WAIT "Press any key to try again"
RETURN
```

وهذا البرنامج عند استدعائه بواسطة الأمر(DO) يقوم بمسح الشاشة ابتداء من السطر (20) ولأسفل . وذلك لكى تظل البيانات موجودة على الشاشة . ثم يقوم بعرض رسالة الخطأ (Message) التى يتم ادخالها مع الأمر (DO) . كما يشغل الجرس (Bell) وينتظر حتى يضغط المستخدم على أى مفتاح أو ينفذ أى شيء يكون موضحا فى الرسالة (Message) . فمثلا عند كتابة الأمر التالى :

```
DO Error WITH "No such customer number!"
```

فان البرنامج يعرض الآتى أسفل الشاشة :

```
No such customer number !
Press any key to try again
```

كما يشغل الجرس (Bell) وينتظر من المستخدم الضغط على أى مفتاح لاستكمال تنفيذ البرنامج الرئيسى .

١٩ - ٤ برنامج التحقق من رقم العميل

هذا هو البرنامج الثالث المستخدم بواسطة برنامج حسابات العملاء ، وهو يتيح للمستخدم استعمال اسم العميل أو رقمه فى البحث عن بيانات عميل محدد . ويتم

ذلك من خلال أمر واحد كالآتي :

```
DO GetCust WITH ;
M_Cust_No,M_Cust_Name,M_Address,Exiting
```

وهذا الأمر يؤدى الى تنفيذ الآتي :

١- عرض شاشة للمستخدم لعرض اسم العميل أو رقمه .
٢- اذا أراد المستخدم الخروج يتم اخبار البرنامج القائم بالاستدعاء بذلك .
٣- البحث عن أرقام الحساب الخاصة بالاسم الذى تم ادخاله وعرضها على الشاشة .
٤- فى حالة ادخال رقم حساب غير موجود يتم عرض رسالة خطأ والسماح للمستخدم بالمحاولة مرة ثانية .
٥- العودة للبرنامج القائم بالاستدعاء مع اعطائه رقم العميل واسمه وعنوانه وعرض هذه البيانات على الشاشة لكى يتحقق المستخدم أن هذا هو العميل المطلوب .

وهذا البرنامج سوف يستخدم كثيرا فى برنامج حسابات العملاء . ولذلك فسوف يتم شرحه بالتفصيل فى نهاية هذا الفصل .

١٩ - ٥ انشاء ملف الخطوات

يتم انشاء ملف الخطوات بنفس طريقة انشاء أى ملف أوامر (Command file) . وذلك باستخدام الأمر (MODIFY COMMAND) ثم كتابة اسم الملف ونسميه فى هذه الحالة (Proclib1.dbf) وهو اختصار (Procedure Library 1) . وملف الخطوات يمكن أن يحتوى على عدد من البرامج بحد أقصى ٣٢ برنامجا . ولكن فى نظام حسابات العملاء فان الملف (Proclib1) يحتوى على ثلاثة برامج فقط كما سبق الايضاح .

وكل برنامج فرعي (Procedure) يبدأ بالأمر (PROCEDURE) يليه اسم البرنامج وينتهي بالأمر (RETURN) . وفي حالة استخدام المعاملات (Parameters) يتم استخدام الأمر (PARAMETERS) فى السطر التالى مباشرة لسطر الأمر (PROCEDURE) .

والسطور التالية توضح أوامر الملف (ProcLib1) .

```
* * * * * * * * * * * * * * *  Proclib1.prg
*    General Procedure for the AR System

* - - - - -  Display screen title.
PROCEDURE Title
PARAMETERS Title
    CLEAR
    @ 1,0  SAY Title
    @ 1,60 SAY DTOC(DATE()) + " " + TIME()
    @ 2,0  SAY Uline
    ?
    ?
RETURN

* - - - - -  Display error message.
PROCEDURE Error
PARAMETERS Message
    @ 20,0
    CLEAR
    @ 20,3 SAY Message
    ? CHR(7)
    WAIT "Press any key to try again"
RETURN

* - - - - -  Look up customer by number or name
PROCEDURE GetCust
PARAMETERS M_Cust_No, M_Name, M_Address , Exiting

    * - - Set up loop for validating customer number.
    * - - Enter customer number or customer name and
    * - - look form it.
    Valid = .F.
    DO WHILE .NOT. Valid

       * - - - Get customer name or number
       Lookup = SPACE(20)
       @ 4,0  CLEAR
```

```
@ 15,5 SAY "Enter customer number or name"
  GET Lookup
@ 17,5 SAY "Press Return to exit"
READ

* - - - If nothing entered , return "exiting"
IF Lookup = "  "
   Exiting = .T.
   RETURN
ENDIF (Lookup is blank)

*  Lookup by name if name entered.
IF ASC(Look up)>= 65
   Lookup = UPPER(TRIM(Lookup)
   SET INDEX TO CustName, CustNo
   SET EXACT OFF
   SEEK Lookup
   * - - - - -  If name found
   IF FOUND()
       * - - - -Display customers with required
       * - - - - -  name
       M_Cust_No = Cust_No
       @ 5,0  CLEAR
       @ 6,0  SAY "Number   Name   Address"
       ?
       DISPLAY OFF WHILE UPPER(Cust_Name) = ;
       Lookup Cust_No , Cust_Name, Address
       @ 22,2 SAY "Enter customer number" GET ;
         M_Cust_No PICTURE "9999"
       READ
       Lookup = STR(M_Cust_No , 4)
   ELSE(If name not found)
       DO Error WITH "Not found"
   ENDIF(name not found)
ENDIF(name entered)
```

```
* - - - - - -  Lookup by customer number.
IF VAL(Lookup) > 0
    M_Cust_No = Val(Lookup)
    SET INDEX TO CustNo, CustName
    SEEK M_Cust_No
    * - - - - - -  If found, Continue, else ask again.
    IF FOUND()
        Valid = .T.
        M_Name = TRIM(Cust_Name)
        M_Address = TRIM(Address)
    ELSE
        DO Error with "Not found"
    ENDIF
 ENDIF (number entered)
ENDDO (while invalid entry)
RETURN
```

١٩ - ٦ فتح ملف الخطوات

حتى يمكن تشغيل أى برنامج داخل ملف الخطوات يلزم فتح هذا الملف . ويتم ذلك باستخدام الأمر (SET PROCEDURE TO) ثم كتابة اسم الملف المطلوب . فمثلا لفتح ملف الخطوات الخاص ببرنامج حسابات العملاء (A/R) ، يتم كتابة السطر التالى :

```
SET PROCEDURE TO ProcLib1
```

وتنفيذ هذا الأمر يؤدى الى تحميل الملف (ProcLib1) فى الذاكرة المؤقتة (RAM) ، وبالتالى يمكن تنفيذ أى برنامج داخل هذا الملف باستخدام الأمر (DO) كما سبق الايضاح .

ولاغلاق ملف الخطوات المفتوح يمكن كتابة الأمر (CLOSE PROCEDURE) . كما يمكن فتح ملف خطوات آخر . حيث أن هذا يؤدى الى اغلاق ملف الخطوات السابق . مع ملاحظة أنه لايمكن فتح أكثر من ملف خطوات فى نفس الوقت .

١٩ - ٧ ادخال المعاملات (Parameters)

يستخدم الأمر (PARAMETERS) في ادخال المعاملات الى برامج الخطوات (Procedures) كما سبق الايضاح ، وهو يؤدى الى ادخال المعاملات التي يتم كتابتها مع الأمر (DO) في المتغيرات الموجودة بعده . ويتم ادخال المعاملات بنفس الترتيب . ويجب ملاحظة أن عدد المعاملات التي يتم ادخالها بعد الأمر (DO) يجب ان يكون مطابقا لعدد المتغيرات التي يتم كتابتها بعد الأمر (PARAMETERS) . والا فان البرنامج يتوقف ويعطى رسالة خطأ .

فمثلا السطور التالية تمثل انشاء برنامج خطوات اسمه (GetArea) يستخدم المتغيرات الثلاثة التالية (Length) ، (Width) ، (Area) .

```
PROCEDURE GetArea
PARAMETERS Length, Width, Area
    Area = Length * Width
RETURN
```

ولتشغيل هذا البرنامج مع ادخال الطول (20) والعرض (25) والمساحة (Area) يتم كتابة الآتى :

```
Area = 0
DO GetArea WITH 20, 25, Area
```

وعند السؤال عن المساحة (Area) يتم كتابة الآتى :

```
? Area
```

فيلاحظ ظهور العدد (500).

ويلاحظ هنا ضرورة تعريف المتغير (Area) قبل استخدامه . ويمكن استخدام أى متغيرات (X, Y, Z) كمعاملات مع تعريفها قبل استخدامها وذلك كالآتى :

```
X = 5
Y = 10
Z = 0
DO GetArea WITH X , Y, Z
```

وعند السؤال عن قيمة المتغير (Z) كالآتى :

```
? Z
```

يلاحظ ظهور العدد (50) .

ويجب ملاحظة أن الأمر (PARAMETERS) لايتم استخدامه فى ادخال المعاملات الى البرنامج فقط ولكنه يستخدم فى اخراج قيم من البرنامج مثل المعامل (Area) مثلا . كما يجب ملاحظة أن ادخال المعاملات لايستخدم فقط مع برامج الخطوات (Procedure) ولكنــه يستخدم أيضا مع أى برنامج يتم انشاؤه بـــواسطة برامج عائلة (DBase) .

١٩ - ٨ دراسة برنامج الخطوات (GetCust)

هذا البرنامج مكتوب فى ملف الخطوات (ProcLib1) السابق شرحه . والبرنامج يبدأ بالسطرين التاليين :

```
PROCEDURE GetCust
PARAMETERS M_Cust_No, M_Name, M_Address , Exiting
```

والمعاملات (Parameters) فى هذا البرنامج تستخدم أساسا لاعادة البيانات المطلوبة الى البرنامج القائم بالاستدعاء .

والبرنامج يقوم بسؤال المستخدم عن الاسم أو الرقم المطلوب البحث عنه . ويقوم بتخزين هذا الاسم أو الرقم فى المتغير (Lookup) ، واذا لم يدخل المستخدم أى اسم فان برنامج الخطوات يعود الى البرنامج المستدعى (Calling Program) مع اعادة القيمة (.T.) فى المتغير (Exiting) . ويتــم ذلك من خلال السطور التالية :

```
IF Lookup = "  "
    Exiting = .T.
    RETURN
ENDIF(lookup is blank)
```

واذا أدخل المستخدم اسم عميل معين فان البرنامج يحول هذا الاسم الى حروف كبيرة (Uppercase) ، ثم يبحث عن هذا الاسم . وذلك من خلال السطور التالية :

```
IF ASC(look up)>= 65
    Lookup = UPPER(TRIM(Lookup))
    SET INDEX TO CustName, CustNo
    SET EXACT OFF
    SEEK Lookup
```

ويتم استخدام الدالة (ASC) لمعرفة ما اذا كان المستخدم أدخل اسما أم رقما ، حيث أن كود الآسكي للحرف (A) هو (65) . أما الأرقام فان الكود الخاص بها يكون أقل من (65) . ولذلك فان تحقق الشرط بعد (IF) يعني أن المستخدم يدخل حروفا وليس أرقاما ، وهذا يعني أنه أدخل الاسم وليس رقم العميل . وفي هذه الحالة يتم فتح ملف الفهرس (CustName) باعتباره الفهرس الرئيسي ، ثم يتم البحث عن هذا الاسم . ويلاحظ هنا استخدام الأمر (SET EXACT OFF) للسماح للمستخدم بادخال أي عدد من الحروف والحصول على كل الأسماء التي تبدأ بهذه الحروف .

وعند العثور على هذا الاسم يتم عرض بيانات جميع الأسماء المشتركة في هذا الاسم حتى يختار منهم المستخدم الاسم المطلوب . وفي هذه الحالة يقوم بادخال رقم الحساب الخاص به ، وهذا يؤدى الى الوصول الى السجل المطلوب .

ويتم ذلك من خلال السطور التالية :

```
IF FOUND()
    * - - - - - -  Display customers with required
    * - - - - - -  name
    M_Cust_No = Cust_No
    @ 5,0  CLEAR
    @ 6,0  SAY "Number   Name   Address"
    ?
    DISPLAY OFF WHILE UPPER(Cust_Name) = ;
       Lookup Cust_No , Cust_Name, Address
    @ 22,2 SAY "Enter customer number" GET ;
       M_Cust_No PICTURE "9999"
    READ
    Lookup = STR(M_Cust_No , 4)
```

وفي حالة عدم العثور على هذا الاسم يتم عرض رسالة خطأ . ويلاحظ هنا أنه تم استخدام برنامج الخطوات (Error.prg) الموجود في نفس الملف . وهذا يعني أنه

يمكن استدعاء برنامج خطوات من برنامج خطوات آخر موجود داخل نفس ملف الخطوات (Procedure File) .

والسطور التالية توضح هذه العملية .

```
    ELSE(If name not found)
       DO Error WITH "Not found"
    ENDIF(name not found)
ENDIF(name entered)
```

والجزء التالى من البرنامج يعالج حالة ادخال المستخدم لرقم العميل دون اسمه . وحيث أن القيم التى تنتج من الدالة (VAL) مع أى سلسلة حرفية تكون صفرا ، فان الشرط الموجود بعد (IF) لايتحقق الا اذا كان مايدخله المستخدم فى المتغير (Lookup) أعدادا وليست حروفا .

وحيث أن الحقل (Cust_No) هو حقل عددى لذلك يجب عند البحث عنه التأكد من أن القيمة الجارى مقارنتها قيمة عددية ، ولذلك تستخدم الدالة (VAL) لتحويل المتغير (Lookup) الى متغير عددى ، ثم يتم البحث عن هذا المتغير بعد فتح الفهرس الخاص برقم العميل (CustNo) . ويتم ذلك من خلال السطور التالية :

```
IF VAL(Lookup) > 0
    M_Cust_No = Val(Lookup)
    SET INDEX TO CustNo, CustName
    SEEK M_Cust_No
```

واذا تم العثور على رقم العميل يتم تغيير المتغير (Valid) الى (.T.) وهذا يؤدى الى عدم تنفيذ الحلقة التكرارية مرة ثانية ، ثم يتم تخزين الاسم (Cust_Name) الخاص بالسجل الذى تم الوصول اليه فى المتغير (M_Name) والعنوان الخاص به (Address) فى المتغير (M_Address) . واذا لم يتم العثور على رقم العميل يتم عرض رسالة خطأ (Error Message) . والسطور التالية توضح هذه العملية :

```
    IF FOUND()
        Valid = .T.
        M_Name = TRIM(Cust_Name)
        M_Address = TRIM(Address)
    ELSE
```

```
        DO Error with "Not found"
    ENDIF(not eof)
ENDIF(number entered)
ENDDO(While invalid entry)
RETURN
```

ويلاحظ خلال هذا البرنامج أن ملف قاعدة البيانات لم يتم فتحه وذلك لأنه يكون قد تم فتحه من خلال البرنامج الرئيسى لنظام حسابات العملاء . كما يجب ملاحظة أن ملف الخطوات يجب فتحه من خلال البرنامج الرئيسى حتى يمكن استخدام البرامج الموجودة فيه فى البرامج الفرعية للنظام .

الفصل العشرون

برامج القائمة الرئيسية والادخال والتعديل



فى هذا الفصل سوف يتم دراسة برنامج القائمة الرئيسية لحسابات العملاء (Main Menu) وبرنامج ادخال عملاء جدد وبرنامج ادخال حركة الصرف وبرنامج ادخال حركة السداد بالاضافة الى برامج التعديل . وحيث أن هذه البرامج لاتختلف عن برامج المخازن السابق شرحها من حيث تصميمها المنطقى ، لذلك فسوف يتم كتابتها دون شرحها مع شرح أى أوامر جديدة لم يسبق استخدامها .

٢٠ - ١ برنامج القائمة الرئيسية

هذا البرنامج لايختلف عن باقي برامج القائمة الرئيسية . حيث أنه يعرض قائمة على المستخدم للاختيار منها . وبناء على اختيار المستخدم يقوم البرنامج بالتفرع الى البرامج الأخرى . ولكن فى هذا البرنامج يتم كتابة الأمر التالى :

```
SET PROCEDURE TO ProcLib1
```

وذلك لفتح ملف الخطوات (Proclib1) السابق شرحه فى الفصل السابق .كما يجب كتابة الأمر (CLOSE PROCEDURE) قبل نهاية البرنامج .

وعند تشغيل هذا البرنامج تظهر الشاشة الموضحة بالشكل (٢٠ - ١) .

```
02/20/90        09:50:30
Accounts Receivable Main Menu

1.  Add new customers
2.  Enter new charges
3.  Enter payments/Adjustments
4.  Print reports
5.  Edit data
6.  Do monthly posting
7.  Exit
             Enter choice (1 - 7 )
```

شكل (٢٠ - ١)

ويتكون هذا البرنامج من السطور التالية :

```
* * * * * * * * * * * * * * *AR.prg
Accounts Receivable Main menu
 ALL

 - - Declare variables as public for passing to
 - - procedures.
: M_Cust_No, M_Name, M_Address, Message, Exiting

 - - - -  Open Procedure file ProcLib1.prg
ROCEDURE TO ProcLib1

 - - - -  If color monitor in use , set colors .
COLOR()
ET COLOR TO GR+/B , W+/RB


 - - - -  Set Parameters
ELL OFF
ELETED ON
EVICE TO SCREEN
EADING OFF
'ATUS OFF
LK OFF

 - - Get the date (RUN needs abount 320KB RAM).

TE

 - - - -  Create underline variable Uline.
= REPLICATE("_" , 80)

 - - - -  Set up a loop for the main menu.
 = 0
LE Choice # 7
```

```
    CLEAR
    * - - - - - - -  Print screen title
    DO Title WITH "Accounts Receivable Main Menu"
    TEXT
        1.    Add new customers
        2.    Enter new charges
        3.    Enter payments / adjustments
        4.    Print reports
        5.    Edit data
        6.    Do monthly posting
        7.    Exit
    ENDTEXT
    @ 23,1 SAY "Enter choice (1 - 7) " ;
       GET Choice PICT "9" RANGE 1,7
    READ
    * - - - - - -   Branch accordingly.
    DO CASE
        CASE Choice = 1
           DO NewCust
        CASE Choice = 2
           DO NewChrg
        CASE Choice = 3
           DO NewPay
        CASE Choice = 4
           DO ARPrint
        CASE Choice = 5
           DO AREdit
        CASE Choice = 6
           DO Post
    ENDCASE
ENDDO(while choice # 7)

* - - - - - -  Close procedure file and exit.
CLOSE PROCEDURE
CLEAR
QUIT
```

ويلاحظ فى هذا البرنامج استخدام الأمر (PUBLIC) مع المتغيرات (M_Cust_No)، (M_Name)، (M_Address)، (Exiting). وذلك حتى يمكن استخدام هذه المتغيرات فى جميع البرامج الفرعية دون الحاجة الى اعادة انشائها من جديد . كما أن هذا يعتبر ضروريا لاستخدام برنامج الخطوات (GetCust) . وحتى لا نحتاج الى اعادة انشاء هذه المتغيرات فى كل مرة يراد فيها استخدام هذا البرنامج .

ويلاحظ أيضا استخدام برنامج الخطوات (Title) لكتابة عنوان الشاشة ورسم خط فيها .

ويلاحظ أيضا استخدام الأمر (SET COLOR) فى تجميل الشاشة وزيادة جاذبيتها ، وهذا الأمر مع المعاملات الموجودة معه يؤدى الى تكوين خلفية زرقاء مع الكتابة باللون الأصفر فى الشاشة الرئيسية ، كما يؤدى الى تكوين خلفية بنفسجية مع الكتابة باللون الأبيض فى الأعمدة الضوئية (Highlights) . ويمكن تعديل هذه الألوان حسب الحاجة .

والأمر (RUN DATE) تم استخدامه لتشغيل برنامج التاريخ (DATE) الموجود فى نظام التشغيل (MS_DOS) حتى يسمح للمستخدم بتعديل التاريخ الحالى حسب الحاجة . ويجب ملاحظة أن استخدام الأمر (RUN) من خلال (DBase III+) يتطلب ذاكرة مؤقتة لاتقل عن ٣٢٠ ك بايت .

٢٠ - ٢ برنامج اضافة العملاء (NewCust.prg)

هذا البرنامج لايختلف كثيرا عن برنامج اضافة أرقام الأجزاء الجديدة فى نظام المخازن السابق شرحه ولكن نظرا لأن نظام حسابات العملاء يعتمد على ربط الملفات، لذلك فان هذا البرنامج يضيف جزءا جديدا لاختبار كل رقم عميل جديد يتم ادخاله والتأكد من عدم تكراره . كما أن البرنامج يقوم بزيادة رقم العميل آليا مع كل اضافة لعميل جديد ويسمح للمستخدم بقبول هذا الرقم أو ادخال رقم آخر جديد .

وعندما يختار المستخدم الرقم (1) من القائمة الرئيسية للبرنامج الرئيسى (AR.prg) ، فان البرنامج يمسح الشاشة ويعرض الرسالة التالية :

```
Enter Customer number (0 to quit) : 1001
```

فى هذه الحالة يستطيع المستخدم الضغط على مفتاح الادخال للموافقة على هذا لرقم أو كتابة صفر (0) للرجوع الى القائمة الرئيسية ، أو كتابة رقم حساب آخر .

وعندما يكتب المستخدم رقما موجودا تظهر الرسالة التالية :

```
Number already in use!
```

ثم يسمح البرنامج للمستخدم بالمحاولة مرة ثانية .

وعندما يكتب المستخدم رقما غير موجود تظهر شاشة الادخال حتى يستطيع المستخدم ادخال باقى بيانات هذا العميل . والشكل (٢٠ - ٢) يوضح شاشة الادخال المستخدمة فى هذا البرنامج ، والتى يتم انشاؤها من خلال قوائم برنامج المساعد (Assistant) او عن طريق كتابة السطرين التاليين من مشيرة النقطة (Dot Prompt).

```
Add New
Customers

Customer Number :    9999
Customer Name : XXXXXXXXXXXXXXXXXX
Address : XXXXXXXXXXXXXXXXXXXXXX  Phone : XXXXXXX

Starting Balance : 99999.99   Terms:            XXXXXXXXXXXX
Balance(30 days): 99999.99    Balance(60 days):  99999.99
Balance(90 days): 99999.99    Balance(90+days):  99999.99
```

شكل (٢٠ - ٢)

```
USE Customer
CREATE SCREEN FNewCust
```

وفى هذه الحالة تظهر قوائم الاختيارات التى يتم عن طريقها تحميل الحقول

المطلوب ظهورها على السبورة (Blackboard) . ثم يتم تحديد أماكن هذه الحقول على الشاشة كما سبق الايضاح فى الكتاب الأول .

وحيث أن رقم العميل يتم التحكم فيه من خلال البرنامج ، لذلك يجب عدم السماح للمستخدم بتعديله . وهذا يتم تنفيذه عند تصميم الشاشة . حيث يتم تحريك مؤشر التصحيح الى العمود الضوئى المقابل لرقم العميل (Customer Number) ثم الضغط على مفتاح (F10) ثم الضغط على مفتاح الادخال عند وقوف المؤشر على الاختيار (ACTION) لتحويله الى (Display/Say) . ثم يتم الضغط على مفتاح (F10) مرة أخرى للرجوع الى السبورة (Blackboard) . ويلاحظ فى هذه الحالة اختفاء العمود الضوئى (Highlight) مع بقاء الأرقام (9999) الدالة على نوع الحقل . وهذا يعنى أن المستخدم يرى رقم العميل ولكن لايمكنه تعديله .

والبرنامج المستخدم فى اضافة العملاء يسمى (NewCust.prg) ويتكون من السطور التالية :

```
* * * * * * * * * * * * * * * * * * * NewCust.prg
*    Add new customers to the A/R system.
*    Called from AR main menu
USE Customer INDEX CustNo, CustName

* - - Store largest customer number to memory variable
* - - Next_No.
GO BOTT
Next_No = Cust_No

* - - - - - -  Print screen title.
DO Title WITH "Add New Customers"

* - - - - - -  Set up loop for adding customers.
Exiting = .F.
DO WHILE .NOT. Exiting
    * - - - - - -  Increment customer number by 1
    Next_No = Next_No + 1
    * - - - Suggest next number , but allow user to
    * - - - change it.
    @ 15,5 SAY "Enter customer number (0 to quit): " ;
      GET Next_No PICT "9999"
```

```
    @ 17,5 SAY "Press Return to accept number "
    READ
    * - - - - If zero entered , return to main menu.
    IF Next_No = 0
        Exiting = .T.
        LOOP
    ENDIF

    * - - - Check to see if number is already taken.
    * - - - If it is,loop and ask for another number.
    SEEK Next_No
    IF FOUND()
        ? "Number already in use" , CHR(7)
        LOOP
    ENDIF

    * - -If next number isn't taken , add new record.
    * - -using FNewCust format screen.
    APPEND BLANK
    REPLACE Cust_No WITH Next_No
    REPLACE Term WITH "Net30"
    REPLACE Last_Updat WITH DATE()
    SET FORMAT To FNewCust
    READ
    SET FORMAT TO
    @ 4,0  CLEAR
ENDDO (While not exiting)
* - - - - - -  End of program
CLOSE DATABASES
RETURN
```

ويلاحظ في هذا البرنامج استخدام المتغير (Next_No) في زيادة رقم السجل بواحد آليا وكذلك في اختبار رقم العميل والتأكد من عدم وجوده في ملف العملاء (Customer.dbf).

٢٠ - ٣ برنامج اضافة حركة الصرف (Newchrg.prg)

يتم تشغيل هذا البرنامج عندما يختار المستخدم الرقم (2) من القائمة الرئيسية للبرنامج (AR.prg) .

ويستخدم هذا البرنامج فى اضافة بيانات حركة الصرف الى ملف قاعدة البيانات الخاص بحركة الصرف للعملاء (Charges.dbf) . ولكتابة هذا البرنامج يلزم أولا انشاء شاشة تعديل بيانات الصرف (FNewChrg.fmt) وذلك بكتابة السطرين التاليين من مشيرة النقطة .

```
USE Charges
CREATE SCREEN FNewChrg
```

ويتم تحميل كل حقول ملف قاعدة البيانات (Charges.dbf) ما عدا حقلى الكمية (Amount) والدفع (Billed) . وكما حدث فى شاشة ادخال العملاء (FNewCust) يتم حماية حقل رقم العميل من التعديل . وذلك باستخدام المفتاح (F10) وتحويل الاختيار (Action) الى (Display/Say) .

وعند الانتهاء من تصميم الشاشة تصبح بالصورة الموضحة بالشكل (٢٠ - ٣) . وهنا يجب ملاحظة أن اسم العميل وعنوانه لم يتم اضافته الى الشاشة . حيث أنه لايكون ضمن حقول ملف حركة الصرف (Charges.dbf) . ولكن فى العادة يحتاج المستخدم الى معرفة اسم العميل وعنوانه أثناء تعديل البيانات الخاصة به . لذلك يتم اضافة متغيرات الذاكرة (M_Name) ، (M_Address) الى شاشة الادخال . ولتنفيذ ذلك يتم استخدام الأمر (MODIFY COMMAND) فى تعديل برنامج تصميم الشاشة وذلك كالآتى :

```
MODIFY COMMAND FNewchrg.fmt
```

ثم يتم كتابة السطرين التاليين :

```
@ 4,33 SAY "Name:" + M_Name
@ 5,33 SAY "Address:" + M_Address
```

مع ملاحظة أن الاحداثيات يمكن تعديلها بناء على شكل الشاشة الذى سبق تصميمه .

Add New Payments

Customer Number : 9999
Check Number : 9999999 **Date :** 99/99/99
Part Number : 99999 **Qty :** 99999.99 **Unit Price:** 99999.99
Description : xxxxxxxxxxxxxxxxxxxxxxxxxxx

Curser Movement by Up , Down , Left , and Right arrow keys	
Insert Mode : Ins	Delete Characters : Del
Save : ^End or ^W	Abandon : ^Q

شكل (٢٠ - ٢)

وحتى يختبر مخطط البرامج الشاشة بعد ادخال هذا التعديل يمكنه كتابة الأوامر التالية من مشيرة النقطة (Dot Prompt) .

```
M_name = "Mohamed"
M_Address = "12 Hegaz street"
USE Charges
SET FORMAT TO FNewChrg
APPEND
```

فاذا ظهرت الشاشة فان هذا يدل على أن ملف التشكيل (Format File) سليم . واذا لم تظهر فان هذا يدل على وجود خطأ معين فى ملف التشكيل . ويلاحظ أنه تم انشاء متغيرات الذاكرة (M_Name) ، (M_Address) قبل اجراء الاختبار وذلك لأن هذه المتغيرات ليست موجودة ضمن حقول ملف حركة الصرف (Charges.dbf) . أما أثناء تنفيذ برنامج الاضافة (NewChrg.prg) فان هذه المتغيرات يتم انشاؤها من خلال برنامج الخطوات (GetCust.prg) الذى سبق شرحه .

ويتم كتابة سطور برنامج الاضافة كالآتى :

```
* * * * * * * * * * * * *NewChrg.prg.
* Add individual charges to the Charges database file.
* Called from AR main menu

* - - - - -  Print the screen title
DO Title WITH "Enter New Charges"
* - - - - -  Open both Charges and Customer databases.
SELECT 1
USE Customer INDEX CustNo , CustName
SELECT 2
USE Charges INDEX ChrgNo

* - - - - -  Set up loop for adding entries.
Exiting = .F.
DO WHILE .NOT. Exiting
    * - - Get customer name or number , and validate.
    SELECT 1
    DO GetCust WITH ;
       M_Cust_No, M_Name, M_Address,Exiting
    * - - - Allow user to enter transaction data using
    * - - - FNewChrg screen.
    IF .NOT. Exiting
        SELECT 2
        APPEND BLANK
        REPLACE Cust_No WITH M_Cust_No
        REPLACE Date WITH DATE()
        REPLACE Billed WITH .F.
        SET FORMAT TO FNewChrg
        READ
        CLOSE FORMAT
        REPLACE Amount WITH Qty * Unit_Price
    ENDIF
    @ 3,0 CLEAR
ENDDO(While adding new transactions)
* - - - - - - Return to main menu.
CLOSE DATABASES
RETURN
```

ملاحظة

اذا أريد اضافة الضريبة الى الكمية (Amount) يمكن اضافة السطر التالى مثلا :

```
Amount = 1,05 * Amount
```

وذلك قبل انتهاء الحلقة التكرارية .

٢٠ - ٤ برنامج اضافة حركة السداد (NewPay.prg)

يتم تشغيل هذا البرنامج عندما يختار المستخدم الرقم (٣) من القائمة الرئيسية للبرنامج (AR.prg) . ويستخدم هذا البرنامج فى اضافة بيانات حركة السداد (Payments) الى ملف قاعدة البيانات الخاص بحركة السداد (Payments.dbf).

وهذا البرنامج يماثل تماما برنامج اضافة حركة الصرف . حيث يتم انشاء شاشة ادخال بنفس الطريقة عن طريق كتابة السطرين التاليين :

```
USE Payments
CREATE SCREEN FNewPay
```

وفى هذه الحالة تظهر قوائم الاختيارات الخاصة برسم الشاشة . ويتم تحميل حقول ملف حركة السداد (Payments.dbf) كلها باستثناء حقل الترحيل (Posted). كما يتم استخدام مفتاح (F10) لتحويل حقل رقم العميل (Cust_No) الى حالة (Display/Say) حتى يصبح محميا من أى تعديل بواسطة المستخدم .

ولاضافة المتغيرين (M_Name) ، (M_Address) الى شاشة الادخال يتم استخدام نفس الطريقة السابقة . وذلك باستخدام الأمر (MODIFY COMMAND) عن طريق كتابة السطر التالى :

```
MODIFY COMMAND FNewPay.FMT
```

ثم يتم اضافة السطرين التاليين الى برنامج رسم الشاشة .

```
@ 5,35 SAY "Name:" + M_Name
@ 6,35 SAY "Address:" + M_Address
```

وعند الانتهاء من تصميم الشاشة تصبح بالصورة الموضحة بالشكل (٢٠ - ٤) .

```
                     Add New Payments

  Customer Number :   9999                  Name :
  Check Number    :  9999999                Address :
  Amount   :        9999999.99
  Description :   xxxxxxxxxxxxxxxxxxxxxxxxxxx

  Curser Movement by Up , Down , Left , and Right arrow keys
  Insert Mode : Ins           |   Delete Characters : Del
  Save : ^End  or  ^W         |   Abandon   :   ^Q
```

شكل (٢٠ - ٤)

ويمكن استخدام هذه الشاشة في اضافة أي حركة سداد (Payments) . كما يمكن اضافة أي حركة ضبط (adjustment) عند ارجاع العميل لأي صنف ويراد اضافته للرصيد . وفي هذه الحالة يتم ملء بيانات الشاشة مع تحديد الكمية التي تمت اعادتها . ثم يتم توضيح سبب رجوع هذه الكمية في حقل الوصف (Discript).

وكما تم بالنسبة لبرنامج اضافة حركة الصرف ، يتم استخدام برنامج الخطوات (GetCust) في البحث عن اسم العميل أو رقمه والتأكد من وجوده في الملف . وكما سوف نلاحظ فان البرنامج يماثل برنامج اضافة حركة الصرف (NewChrg.prg) مع اختلافات قليلة جدا . لذلك يمكن نسخه باستخدام الأمر (MODIFY COMMAND) وقراءة نفس هذا الملف في الملف الجديد . وذلك بكتابة (^KR) ثم اجراء التعديلات المطلوبة فيه . وذلك بدلا من كتابته من جديد . والسطور التالية توضح أوامر هذا البرنامج .

```
* * * * * * * * * * * * *  NewPay.prg.
*    Add individual payments to the payments file.
*    Called from AR main menu

* - -Open both Payments and Customer databases.
SELECT 1
USE Customer INDEX CustNo , CustName
SELECT 2
USE Payments INDEX PayNo

* - - - - - -  Set up loop for adding entries.
Exiting = .F.
DO WHILE .NOT. Exiting
    * - - Get customer name or number , and validate.
    SELECT 1
    DO GetCust WITH ;
       M_Cust_No , M_Name , M_Address , Exiting
    * - Allow user to enter transaction data using
    * - FNewPay screen.
    IF .NOT. Exiting
        SELECT 2
        APPEND BLANK
        REPLACE Cust_No WITH M_Cust_No
        REPLACE Date WITH DATE()
        REPLACE Discript WITH "Payment"
        REPLACE Posted WIHT .F.
        SET FORMAT TO FNewPay
        READ
        CLOSE FORMAT
    ENDIF
    @ 3,0  CLEAR
ENDDO (While adding new transactions)
* - - - - - - Return to main menu.
CLOSE DATABASES
RETURN
```

٢٠ - ٥ برنامج التعديل (AREdit.prg)

يتم تعديل حسابات العملاء باستخدام برنامج تعديل اضافة العملاء (EdCust.prg) ، وبرنامج تعديل حركة الصرف (EdChrg.prg) ، وبرنامج تعديل حركة السداد (EdPay.prg) . ويتم التحكم فى هذه البرامج عن طريق برنامج قائمة التعديل (AREdit.prg) . ويتم تشغيل هذا البرنامج عندما يختار المستخدم الرقم (5) من القائمة الرئيسية للبرنامج (AR.prg) . وعند تشغيل هذا البرنامج تظهر القائمة الموضحة بالشكل (٢٠ - ٥) .

```
02/20/90                    09:50:40
Accounts Receivable Edit Menu

    1.  Edit Customer File
    2.  Edit Current Charges
    3.  Edit Current Payments
    4.  Return to Main Menu

    Enter choice (1 - 4 )
```

شكل (٢٠ - ٥)

كما يتم كتابة هذا البرنامج كالآتى :

```
* * * * * * * * * * * * * * *AREdit.prg
*     Menu for editing the A/R System.
*     Called from AR main menu
EChoice = 0
DO WHILE EChoice # 4
     CLEAR
     DO Title WITH "Accounts Receivable Edit Menu"
     TEXT
          1.   Edit Customer File
          2.   Edit Current Charges
          3.   Edit Current Payments
          4.   Return to main menu
     ENDTEXT
     @ 24,3 SAY "Enter choice (1 - 4) " ;
        GET EChoice PICT "9" RANGE 1,4
     READ

     * - - - - -  Branch accordingly.
     DO CASE
          CASE EChoice = 1
               DO EdCust
          CASE EChoice = 2
               DO EdChrg
          CASE EChoice = 3
               DO EdPay
     ENDCASE
ENDDO (Echoice # 4)
* - - - - -  Return to main menu
RETURN
```

٢٠ - ٥ - ١ تعديل ملف العميل (EdCust.prg)

شاشة تعديل ملف العميل (FEdCust.scr) هي نفس شاشة ادخال العملاء (FNewCust.scr) مع اختلاف وحيد وهو أن العنوان يجب تغييره الى (Edit Customers) . ولذلك يتم نسخ ملف شاشة العميل فى الملف الجديد (FEdCust.scr) وذلك كالآتى :

```
COPY FILE FNewCust.scr TO FEdCust.scr
```

ثم يتم كتابة السطر التالى :

```
MODIFY SCREEN FEdCust
```

فى هذه الحالة تظهر السبورة (Blackboard) ويتم تعديل العنوان من (Add New Customers) الى العنوان (Edit Customers) ، ثم تخزين الشاشة الجديدة . أنظر الشكل (٢٠ - ٦) .

Edit Customers

Customer Number : 9999
Customer Name : XXXXXXXXXXXXXXXXXXX
Address : XXXXXXXXXXXXXXXXXXXXXXXX Phone : XXXXXXX

Starting Balance : 99999.99 Terms: XXXXXXXXXXXX
Balance(30 days): 99999.99 Balance(60 days) : 99999.99
Balance(90 days): 99999.99 Balance(90 +days): 99999.99

شكل (٢٠ - ٦)

وبرنامج تعديل ملف العميل يسمى (EdCust.prg) وهو يستخدم برنامج الخطوات (GetCust.prg) فى البحث عن العميل برقمه أو باسمه مثل البرامج السابقة . ويتم كتابة سطور البرنامج كالآتى :

```
* * * * * * * * * * * * * * * * * EdCust.prg
*    Edit Customers information
*    Called from A/R Edit menu
* - - - - -Print the screen title.
DO Title WITH "Enter Customers File"
USE Customer INDEX CustNo, CustName
* - - - - - -  Set up loop for editing.
Exiting = .F.
SET DELETED OFF
DO WHILE .NOT. Exiting
    * - - - - - -  GET Customer by name or number
    DO GetCust WITH ;
      M_Cust_No , M_Name , M_Address , Exiting
    * - Edit using the FEdCust screen (if not exiting)
    IF .NOT. Exiting
        SEEK M_Cust_No
        SET FORMAT TO FEdCust
        EDIT
        SET FORMAT TO
        @ 4,0  CLEAR
    ENDIF
ENDDO (While not exiting)
* - - - - - -  Return to main menu
SET DELETED ON
CLOSE DATABASES
RETURN
```

٢٠-٥-٢ تعديل ملف الصرف (EdChrg.prg)

برنامج تعديل ملف الصرف (Charges.dbf) يسمح للمستخدم بادخال رقم العميل أو اسمه لتحديد مكان هذا العميل في الملف ، ثم استخدام مفتاحي (PgDn) ، (PgUp) في عرض كل حركة خاصة بهذا العميل للوصول الى الحركة المطلوب تعديلها . وتستخدم الشاشة (FEdChrg.scr) في تعديل حركة الصرف . ولانشاء هذه الشاشة يتم نسخ شاشة اضافة الصرف (FNewchrg.scr) في شاشة التعديل (FEdchrg.scr) وذلك بكتابة السطر التالي :

```
COPY FILE FNewChrg.scr TO FEdChrg.scr
```

ثم يتم كتابة السطر التالى :

```
MODIFY SCREEN FEdChrg
```

حتى يتم عـــرض السبورة (Blackboard) وتعديل عنوان الشاشة الى (Edit Charges) كما يتم اضافة تعليمات أسفل الشاشة لتوضح للمستخدم كيفية الانتقال من حركة الى حركة أخرى باستخدام مفتاحى (PgUp) ، (PgDn) .

وحـتى يظهر اسم العميل وعنوانه مع كل حركة يلزم فـتح ملف العملاء (Customer.dbf) ، وربطه بملف الصرف ، وتثبيت مؤشر السـجلات (Record Pointer) على السجل الخاص بهذا العميل ثم الحصول على اسم هذا العميل وعنوانه . ويتم ذلك عن طريق تعديل ملف الشاشة بواسطة الأمر :

```
MODIFY COMMAND FEdchrg.fmt
```

ثم يتم كتابة السطرين التاليين فى هذا الملف

```
@ 4,33 SAY "Name:" + TRIM(A-> Cust_Name)
@ 5,33 SAY "Address:" + TRIM(A-> Address)
```

ويجب ملاحظة أن الاحداثيات الخاصة بهذين السطرين تتوقف على تصميم الشاشة . والشكل (٢٠ - ٧) يوضح صورة الشاشة المستخدمة فى تعديل حركة الصرف .

ولكى يتم التحكم فى مؤشر السجلات (Record Pointer) الموجود فى ملف العملاء للحصول على اسم العميل وعنوانه يجب انشاء علاقة بين ملفى العملاء (Cust.dbf) والصرف (Charges.dbf) وذلك من خلال برنامج تعديل ملف الصرف (EdChrg.prg) ويتضح ذلك من السطور التالية :

Edit Charges			
Customer Number : 9999		Name :	
		Address :	
Invoice Number : 9999999		Date : 99/99/99	
Part Number : 99999	Qty : 99999.99	Unit Price: 99999.99	
Description : xxxxxxxxxxxxxxxxxxxxxxxxxxxx			
Curser Movement by Up , Down , Left , and Right arrow keys			
Insert Mode : Ins		Delete Characters : Del	
Save : ^End or ^W		Abandon : ^Q	

شكل (٢٠ - ٧)

```
* - - - - - Open both Charges and Customer databases.
SELECT 1
USE Customer INDEX CustNo, CustName
SELECT 2
USE Charges INDEX ChrgNo
* - - - - - Set up relationship
SET RELATION TO Cust_No INTO Customer
```

وهذا الربط يجعل المؤشر يقف دائما على نفس السجل الخاص بالعميل بناء على رقم العميل بصرف النظر عن الحركة التي يتم عرضها على الشاشة . وبالتالى يظل اسم العميل وعنوانه معروضا على الشاشة أثناء تعديل أى حركة خاصة بهذا العميل .

والبرنامج يتم كتابة سطوره كالآتى :

```
* * * * * * * * * * * * * * * EdChrg.prg
*    Edit invalid charges in the charges file.
*    Called from A/R Edit menu

* - -Print the screen title.
DO Title WITH "Edit Current Charges"
* - - - - -  Open both Customer and charges databases.
SELECT 1
USE Customer INDEX CustNo, CustName
SELECT 2
USE Charges INDEX ChrgNo
* - - - - -  Set up relationship
SET RELATION TO Cust_No INTO Customer
* - - - Set up memory variables and loop for editing
SET DELETED OFF
Exiting = .F.
DO WHILE .NOT. Exiting
* - - - - -  Get Customer by name or number.
SELECT 1
DO GetCust WITH M_Cust_No, M_Name, M_Address, Exiting
   * - Edit the transaction, if valid and not exiting
   IF .NOT. Exiting
      SELECT 2
      SEEK M_Cust_No
      IF FOUND .AND. Billed
         SET FORMAT TO FEdChrg
         EDIT
         CLOSE FORMAT
      ELSE
        DO Error WITH "Already posted,Make adjustment"
      ENDIF
   ENDIF (not exiting)
   @ 4,0  CLEAR
ENDDO (While not exiting)
* - - - - -  Recalculate amount field.
SELECT 2
REPLACE ALL Amount WITH Qty * Unit_Price
SET DELETED ON
```

```
CLOSE DATABASES
* - - - - -  Return to main menu.
RETURN
```

ويلاحظ من البرنامج استخدام الجملة

```
IF FOUND() .AND. .NOT. Billed
```

حيث أنها تمنع تعديل السجل بعد ترحيله الى الملف الرئيسي . أما اذا كان قد تم ترحيله فتظهر رسالة خطأ (Error Message) توجه المستخدم الى التعديل عن طريق اضافة حركة جديدة الى ملف حركة الصرف ، كما يلاحظ بعد انتهاء عملية التعديل اضافة السطر التالى :

```
REPLACE ALL Amound WITH Qty * Unit_Price
```

وهذا لضمان أن أى تعديل فى الكمية المصروفة أو فى سعرها يتم ادخاله مباشرة فى كمية الصرف الكلية لهذا العميل .

٢٠ - ٥ - ٣ تعديل ملف السداد (EdPay.prg)

هذا البرنامج يماثل تعديل ملف الصرف تماما مع بعض الاختلافات البسيطة ، وهو يستخدم الشاشة (FEdPay.scr) فى التعديل ، ويتم انشاؤها بنسخ الشاشة (FNewPay.scr) وذلك كالآتى :

```
COPY FILE FNewPay.scr TO FEdPay.scr
```

ثم يتم استخدام الأمر (MODIFY SCREEN) فى تحويل العنوان الى (Edit Payments) واضافة الارشادات الخاصة باستخدام مفتاحى (PgUp) ، (PgDn) فى الانتقال من حركة الى أخرى .

كما يتم استخدام الأمر (MODIFY COMMAND FEdpay.fmt) فى ادخال اسم العميل وعنوانه فى الشاشة . وذلك عن طريق كتابة السطرين التاليين فى ملف الشاشة (FEdpay.fmt) :

```
@ 5,35 SAY "Name :" + TRIM(A -> Cust_Name)
@ 6,35 SAY "Address:" + TRIM(A -> Address)
```

والشكل (٢٠ - ٨) يوضح صورة شاشة الادخال .

Edit Payments	
Customer Number : 9999	Name :
Check Number : 9999999	Address :
Amount : 9999999.99	Date : 99/99/99
Description : xxxxxxxxxxxxxxxxxxxxxxxxxx	
Curser Movement by Up , Down , Left , and Right arrow keys	
Insert Mode : Ins	Delete Characters : Del
Save : ^End or ^W	Abandon : ^Q

شكل (٢٠ - ٨)

وبرنامج التعديل يتكون من السطور التالية :

```
* * * * * * * * * * * * * * *EdPay.prg
*    Edit invalid payments in the payments file.
*    Called from AR Edit menu
* - - - - -  Print the screen title.
DO Title WITH "Edit Current Payments"
* - - - Open both Customer and payments databases.
SELECT 1
USE Customer INDEX CustNo, CustName
SELECT 2
USE Payments INDEX PayNo

* - - - - -  Set up relationship
SET RELATION TO Cust_No INTO Customer
```

```
* - - - - - -  Set up loop for editing entries
SET DELETED OFF
Exiting = .F.
DO WHILE .NOT. Exiting
    * - - - - - -  Get Customer by name or number
    SELECT 1
    DO GetCust WITH ;
       M_Cust_No, M_Name, M_Address, Exiting
    * - - If not exiting , and transaction not already
    * - - - - - -  posted , proceed with edit.
    IF .NOT. Exiting
      SELECT 2
      SEEK M_Cust_No
      IF FOUND() .AND. .NOT. Posted
         SET FORMAT TO FEdPay
         EDIT
         CLOSE FORMAT
      ELSE
         DO Error WITH ;
           "Already posted, Make adjustment:"
      ENDIF
    ENDIF (not exiting)
    @ 4,0  CLEAR
ENDDO(while not exiting)
* - - - - - -  Close databases and return to edit menu
SET DELETED ON
CLOSE DATABASES
RETURN
```



الفصل الحادي والعشرون

تقارير برنامج حسابات العملاء



يعتمد برنامج حسابات العملاء (A/R) على مجموعة من التقارير (Reports). مثل الفواتير (Invoices) والتقارير الشهرية والتقارير المختصرة (Summary reports) والتقارير الزمنية (Aging reports) التى توفر للمستخدم المتابعة الدقيقة للبيانات وتقديم الخدمة السريعة للعميل . وكذلك التقارير التاريخية (Historical Reports) التى تساعده على اتخاذ القرارات واجراء التعديلات المطلوبة فى النظام بناء على دراسات احصائية للمخرجات .

وفى الواقع فان كتابة البرامج التى توفر هذه الأنواع من التقارير تحتاج الى كثير من الوسائل المتقدمة فى كتابة البرامج وخصوصا عندما يراد الاحتفاظ بسرعة تشغيل البرنامج وكفاءته . ولذلك فسوف يتم استخدام برامج الخطوات (Procedures) والمعاملات (Parameters) فى بعض البرامج كما سيتم تصميم برامج لأنواع مختلفة من التقارير .

٢١ - ١ برنامج قائمة التقارير الرئيسية (ARPrint.prg)

عندما يختار المستخدم الرقم (4) من القائمة الرئيسية لبرنامج حسابات العملاء (A/R) تظهر شاشة الاختيارات الموضحة بالشكل (٢١ - ١) .

A/R Print Menu

1- Print monthly statements
2- Print monthly summary
3- Print aging reports
4- Do quick looking of customer status
5- Review history
6- Return to main menu

Enter choice (1 - 6)

شكل (٢١ - ١)

والاختيار الأول يستخدم عندما يراد طباعة الفواتير (Invoices) . ويجب تشغيله مرة واحدة في الشهر عندما يراد ترحيل الحسابات الى الملف الرئيسي .

والاختياران (2 , 3) يستخدمان أيضا مرة واحدة في الشهر بعد طباعة الفواتير للحصول على تقرير مختصر عن موقف الفواتير الخاصة بكل عميل .

والاختيار رقم (4) يستخدم عندما يريد العميل مراجعة الحساب الخاص به .

والاختيار رقم (5) يوفر مراجعة أي بيانات تاريخية لأي حساب لمساعدة المستخدم على متابعة السير التاريخي لأي عميل ، وتوفير المعلومات التي تتعلق بفترة معينة محصورة بين تاريخين ، وكذلك توفير معلومات عن حالة بيع صنف معين .

والبرنامج الخاص بعرض القائمة الرئيسية للتقارير (ARPrint.prg) لايختلف عن أي برنامج من برامج القائمة الرئيسية الأخرى . ويتكون من السطور التالية :

```
* * * * * * * * * * * * * * *ARPrint.prg

*    Menu of Print options for the A/R system.
*    Called from A/R main menu

PChoice = 0
SET DELETED ON
DO WHILE PChoice # 6
    CLEAR
    DO Title WITH "A/R Print Menu"
    TEXT
        1.  Print monthly statements
        2.  Print monthly Summary
        3.  Print aging reports
        4.  Do quick looking of customer status
        5.  Review history
        6.  Return to main menu
    ENDTEXT
    @ 24,1 SAY "Enter choice (1 - 6)" ;
      GET Pchoice PICT "9" RANGE 1,6
    READ

* - - - - -   Branch accordingly.
```

```
    DO CASE
        CASE PChoice = 1
            DO Bills
        CASE PChoice = 2
            RepForm = "ARSumm"
            DO AgeSumm
        CASE PChoice = 3
            RepForm = "Aging"
            DO AgeSumm
        CASE PChoice = 4
            DO ARStat
        CASE PChoice = 5
            DO ARHist
    ENDCASE
ENDDO(PChoce # 6)
* - - - - -  Return to main menu
SET DELETED OFF
RETURN
```

ويلاحظ من هذا البرنامج أن الاختيار (2) والاختيار (3) يؤديان الى تنفيذ نفس البرنامج (AgeSumm) ولكن البرنامج يعطى تقريرا مختلفا فى كل حالة كما سيتم الايضاح فيما بعد .

٢١ - ٢ ملف الخطوات (BillProc.prg)

يلاحظ من البرنامج السابق الخاص بقائمة التقارير وجود برنامج الفواتير الشهرية (Bills.prg) الخاص بالاختيار (1) ، وبرنامج تحديد الحالة (ARStat.prg) الخاص بالاختيار (4) ، وهما يستخدمان فى طباعة الفواتير (Invoices) . ولزيادة سرعة تشغيل البرنامج وكفاءته يقوم كل من هذين البرنامجين بتشغيل ملف خطوات . وهذا الملف يتم تسميته (BillProc.prg). وهو يحتوى على برنامج الخطوات (PrintBills.prg) بالاضافة الى برنامج آخر يسمى (RowCheck) وهو يقوم باختبار رقم السطر فاذا زاد عدد السطور عن طول الصفحة يتم الانتقال الى الصفحة التالية . وقد كان يمكن كتابة هذه البرامج كلها فى ملف الخطوات (Proclib1) السابق انشاؤه ولكن ذلك سوف يتطلب استخدام معالج كلمات آخر غير معالج الكلمات المستخدم مع (+DBase III) . وليست هناك حاجة لهذا فى برنامج حسابات العملاء حيث يمكن فتح ملف الخطوات المطلوب وقت الحاجة ثم اغلاقه بعد ذلك . ولذلك فان تعدد ملفات الخطوات لايؤثر فى كفاءة البرنامج .

والملف (BillProc.prg) يتكون من السطور التالية :

```
* * * * * * * * * * * * * * * * * *BillProc.prg
*  Print a bill,using procedures PrintBills and
*  Rowcheck Called from Bills.prg and ARstat.prg
PROCEDURE PrintBills
PARAMETERS M_Cust_No , Printer, Status

* - - - IF printer , print address, date in English
CLEAR
Page = 1
IF Printer
   EngDate = CMONTH (DATE()) + STR(DAY(DATE()), 3) + ;
      "," + STR(YEAR(DATE()), 4)
   @ 1,0  SAY "My company , inc."
   @ 1,60 SAY EngDate
   ROW = 6
ELSE
   ROW = 1
ENDIF

* - - - -Print Customer name and address Using
* - - - -Row variable to control display and eject.
SELECT 1
IF Cus_Name  # "  "
    @ Row,0 SAY Cus_Name
ENDIF
@ Row+1,0 SAY Address
Row = Row + 4

* - - -Print customer number , terms and statrting
* - - -balance from the customer file .
@ Row,0 SAY "Customer No, : " + STR(Cust_No, 4)
@ Row+1,0 SAY "Terms  :" + Terms
@ Row+2,0 SAY "Balance of " + DTOC(LAST_UPDAT) + ":"
@ Row+2,25 SAY Start_Bal PICT "999,999.99"
@ Row+3,0 SAY Uline
Start = Start_Bal
Row = Row + 5

* - - - - - Print heading for charges .
@ Row,0 SAY "Inv. #  Part  Description    QTY"
@ Row,43 SAY "Price   Total   Date "
```

```
Row = Row + 2
* - - Select charges database :list and total current
* - - charges.
SELECT 2
SEEK M_Cust_No
Tot_Charge = 0
DO WHILE Cust_No = M_Cust_No .AND. .NOT. EOF()
    IF Status .OR. .NOT. Billed
       @ ROW,0 SAY Invoice_No
       @ ROW,7 SAY Part_No
       @ ROW,13 SAY Descript
       @ ROW,34 SAY Qty
       @ ROW,39 SAY Unit_Pric PICT "999,999.99"
       @ ROW,50 SAY Amount PICT "999,999.99"
       @ ROW,62 SAY Date
       Tot_Charge = Tot_Charge + Amount
       ROW = ROW + 1
       * - - - - Make as billed if not a status check
       IF .NOT. STATUS
          REPLACE Billed WITH .T.
       ENDIF
    ENDIF(not status report and not already billed)
    SKIP
ENDDO (Cust_No = M_Cust_No)
* - - - - - Print payments heading .
@ ROW+1,0 SAY Uline
@ ROW+2,1 SAY "Payments / Adjustments"
ROW = ROW + 4

* - - - Check row position if displayed on screen.
DO RowCheck WITH 20,64
* - - - Select Payments file : list and total
* - - - payments/ adjustments.
SELECT 3
SEEK M_Cust_No
Tot_Pay = 0
DO WHILE Cust_No = M_Cust_No .AND. .NOT. EOF()
    IF Status .OR. .NOT. Posted
       @ Row,0 SAY "Check #"
       @ Row,9 SAY Check_No
       @ Row,16 SAY Descript
       @ Row,50 SAY Amount PICT "999,999.99"
       @ Row,62 SAY Date
       Tot_Pay = Tot_Pay + Amount
```

```
        ROW = ROW + 1
        * - - Check row position if displayed on screen
        DO RowCheck WITH 20,64
        * - - - Mark as billed if not status report
        IF .NOT. Status
           REPLACE Posted WITH .T.
        ENDIF
    ENDIF(not status report and not already posted)
    SKIP
ENDDO(while Cust_No = M_Cust_No)

* - - - Check row position if displayed on screen.
@ Row,0 SAY Uline
DO RowCheck WITH 17,56

* - - Print starting balance ,total charges, payments,
* - - ending balance , and thank you note.
SET FIXED ON
@ Row+1,5 SAY "Previous balance :"
@ Row+1,25 SAY START PICT "999,999.99"
@ Row+2,5  SAY "Total charges :"
@ Row+2,25 SAY Tot_Charge PICT "999,999.99"
@ Row+3,5  SAY "Payments received"
@ Row+3,25 SAY Tot_Pay PICT "999,999.99"
@ Row+4,5  SAY "Balance due    :"
@ Row+4,25 SAY (Start + Tot_Charge) - Tot_Pay ;
  PICT "999,999.99"
SET FIXED OFF
IF Printer .AND. Status
    @ Row+8,10 SAY * * *  Duplicate Invoice * * *
ENDIF
IF Pritner .AND. .NOT. Status
    @ Row+8,5  SAY "Thank you"
ENDIF
* - - - - -  Pause if not going to the printer.
IF .NOT. Printer
    @ 22,0 CLEAR
    WAIT
ENDIF

* - - - - -  Done printing bill, Return to menu.
RETURN
```

```
* - - - Procedure for checking row positions on screen
* - - - or printer
PROCEDURE RowCheck
PARAMETERS ScreenMax, PrintMax
IF .NOT. Printer .AND. Row >= ScreenMax
    @ 23,0 CLEAR
    WAIT "Press any key for next page"
    Row = 1
    CLEAR
ENDIF(Row too big for screen)
IF Printer .AND. Row >= PrintMax
    @ Row+2,70 SAY "Page" + STR(Page, 1)
    Page = Page + 1
    EJECT
    ROW = 5
ENDIF (Row too big for printer)
RETURN
```

ويلاحظ فى البرنامج استخدام متغير الذاكرة (Status) لطباعة تقارير الحالة (Status) دون أن يؤثر ذلك على الفاتورة الشهرية . فاذا تم ادخال القيمة (.T.) الى المتغير (Status) فان هذا يعني أن المطلوب تقرير حالة فقط وليس تقرير شهرى، ولذلك تتم طباعة التقرير دون ترحيل . أما اذا كان التقرير فاتورة شهرية (Not Status) فيتم طباعته مع تحويل حقل الترحيل الى (False) حتى يتم ترحيله بعد ذلك باستخدام الاختيار (6) من القائمة الرئيسية لبرنامج حسابات العملاء .

ويجدر العلم أن الملف السابق ما هو الا ملف خطوات يتم استدعاؤه بواسطة برنامج التقارير الشهرية (Bills.prg) وبرنامج تحديد الحالة (ARStat.prg) . وهذان البرنامجان سيتم شرحهما فى الأجزاء التالية .

٢١ - ٣ برنامج الفواتير الشهرية (Bills.prg)

يستخدم هذا البرنامج عند ادخال الاختيار رقم (1) فى قائمة برنامج طباعة تقارير حسابات العملاء . وهذا البرنامج يقوم بطباعة الفواتير الشهرية وفى نفس الوقت يقوم بادخال القيمة (.F.) فى حقل الترحيل سواء كان حقل التسديد (Billed) فى ملف الصرف (Charges.dbf) ، أو حقل التسديد (Posted) فى ملف السداد (Payments.dbf) . وذلك حتى يتسنى للمستخدم بعد ذلك

ترحيل هـــذه الفـــواتير الى الملف الرئيسي عن طريق برنامج الترحيل الشهري (Monthly Posting) بالاختيار رقم (٦) من القائمة الرئيسية لبرنامج حسابات العملاء .

ويتكون هذا البرنامج من السطور التالية :

```
* * * * * * * * * * * * * * * * *Bills.prg

*     Prints monthly statemetns.
*     Called from AR Print menu.

* - - - - -  Have user prepare printer (or cancel).
Proceed = "  "
DO Title WITH "Print Monthly Bills "
@ 15,5 SAY "Repare printer and press a key to proceed"
@ 17,5 SAY "(Type X to cancel)" GET Proceed PICT "!"
READ
* - - - - -  Return to menu if requested.
IF Proceed = "X"
    RETURN
ENDIF

* - - - Open files and delete records with "0" Amount.
SELECT 1
USE Customer INDEX CustNo
SELECT 2
USE Charges INDEX ChargNo
DELETE ALL FOR Amount = 0
SELECT 3
USE Payments INDEX PayNo
DELETE ALL FOR Amount = 0 ,

* - - - Set decimal place to 2 , send @ ... SAY to
* - - - printer and open bill_printing procedure file.
CLEAR
SET DECIMALS TO 2
SET DEVICE TO PRINT
SET PROCEDURE TO BillProc

* - Set parameters to printer and "not status" report.
```

```
Printer = .T.
Status = .F.

* - - - Loop through Customer database and print a
* - - - bill for every one
SELECT 1
DO WHILE .NOT. EOF()
    Lookup = Cust_No
    DO PrintBills WITH Lookup , printer , status
    EJECT

    * - - - - - -  Set next customer
    SELECT 1
    SKIP
ENDDO

* - - - - - -  Done , Close files
SET DEVICE TO SCREEN
CLOSE DATABASES
CLOSE PROCEDURE
* - - - - - -  Open ProcLib1 procedure file
SET PROCEDURE TO ProcLib1
* - - -  Print reminder about posting , then return to
* - - -  main menu .
CLEAR
TEXT
      monthly postings (main menu option 5) should be
      performed immediately after printing the monthly
      statements.
ENDTEXT
?

@ 22,10 SAY "Press any key to return to main menu..."
WAIT " "
RETURN TO Master
```

ويلاحظ فى هذا البرنامج استخدام ملف الخطوات (BillProc.prg) لطباعة التقارير الشهرية . كما يلاحظ فتح ملف الخطوات الآخر (ProcLib1) قبل نهاية البرنامج حتى يستخدم فى باقى البرامج التى سوف تحتاجه ؛ كما يلاحظ استخدام الأمر (RETURN TO MASTER) للرجوع الى القائمة الرئيسية لبرنامج حسابات العملاء مباشرة حتى يتسنى للمستخدم استخدام الاختيار رقم (6) فى ترحيل الفواتير .

٢١ - ٤ برنامج اختبار الحالة (ARStat.prg)

يستخدم هذا البرنامج عند ادخال الاختيار رقم (٤) في قائمة برنامج طباعة تقارير حسابات العملاء ، وهو يسمح للمستخدم باختبار حالة العميل من حيث تسديد الفواتير (Billed) او عدم تسديدها ، وهو يتيح له الحصول على صورة أخرى من الفاتورة المرسلة الى العميل لتسديدها . ويلاحظ في هذا البرنامج اعطاء المتغير (Status) القيمة (True) حتى يعرف برنامج الخطوات (PrintBills) أن المطلوب هو اختبار الحالة فقط وليس الفاتورة الحقيقية . وهذا البرنامج يسمى (ARStat.prg) ويتكون من السطور التالية :

```
* * *  * * * * * * * * * * * *ARStat.prg
*    Quick lookup of a single statement.
*    Called from A/R Print menu.

* - - Open files and delete records with "0" amounts.
SELECT 1
USE Customer INDEX CustNo
SELECT 2
USE Charges INDEX ChrgNo
DELETE ALL FOR Amount = 0
SELECT 3
USE Payments INDEX PayNo
DELETE ALL FOR Amount = 0

* - - - Set up memory variables for status report.
Status = .T.
Printer = .F.
M_Cust_No = 0
M_Name = "  "
Exiting = .F.
DO WHILE .NOT. Exiting
    * - - - - - -  Print screen title
    DO Title WITH "Quick lookup of current status"
    * - - - - - -  Get Customer by number or name
    SELECT 1
    DO GetCust WITH ;
      M_Cust_No , M_Name, M_Address, Exiting
    * - - - - - -  Proceed with bill.
    IF .NOT. Exiting
        * - - - - - -  Ask about printer.
        @ 5,0  CLEAR
        LP = "  "
```

```
@ 15,5 SAY "Send statement to printer ? Y/N ;
  GET LP PICT "!"
READ
CLEAR
* - - - - -  Set up printer if necessary
IF LP = "Y"
    Printer = .T.
    SET DEVICE TO PRINT
ENDIF

* - - -Print current statement for customer.
SET PROCEDURE TO BillProc
DO PrintBill WITH M_Cust_No , Printer, STATUS
CLOSE PROCEDURE
SET PROCEDURE TO ProcLib1
* - - - - -  turn off printer
IF Printer
    EJECT
    SET DEVICE TO SCREEN
ENDIF
ENDIF (not exiting)
* - - - Close files and return to main menu
CLOSE DATABASES
RETURN
```

٢١ - ٥ برنامج التقارير المختصرة والزمنية (AgeSumm.prg)

الاختياران (2) ، (3) من قائمة برنامج طباعة تقارير حسابات العملاء يسمحان للمستخدم بالحصول على تقارير شهرية مختصرة وتقارير زمنية (Aging Reports). وهذه التقارير يتم انشاؤها عادة بعد طباعة الفواتير الشهرية مباشرة .

والتقارير الشهرية المختصرة (Summary Reports) تعرض الموازنة الحالية للعميل وموقف الصرف والسداد الخاص به . حيث يتم عرض تقرير بالصورة الموضحة بالشكل (٢١ - ٢) .

ويتم انشاء صورة التقرير باستخدام الأمر (CREATE REPORT) وذلك كالآتي :

```
USE Customer
CREATE REPORT ARSumm
```

Page No. 1
11/21/90

Monthly Activity Summary

Cust. No.	Name	Current Balance	Current Charges	Current Payments	Last Update
1000	Ahmed Salem	80.00	180.00	100.00	02/30/90
1021	Tarek Fathy	0.00	0.00	150.00	01/10/90
* * Total * *		80.00	180.00	250.00	

شكل (٢١ - ٢)

وفى هذه الحالة تظهر قوائم برنامج المساعد (Assistant) التى يتم من خلالها تحديد محتويات الأعمدة (Columns). وذلك كما هو موضح بالشكل (٢١ - ٣) .

	Contents	Heading	Width	Decimal	Total
1	Cust_No	Cust ; No	5	0	N
2	Trim(Csut_Name)	Name	26	0	
3	Start_Bal	Current ; Balance	7	2	Y
4	Chg_Curr	Current ; Charge	8	2	Y
5	Pay_Curr	Current ; Payments	8	2	Y
6	Last_Updat	Last ; Posted	8	0	

شكل (٢١ - ٣)

أما التقارير الزمنية فتسمح للمستخدم بعرض الموازنة الخاصة بكل عميل في فترات مختلفة لمراقبة موقف هذا العميل ، حيث يكون التقرير بالصورة الموضحة بالشكل (٢١ - ٤) .

Page No. 1
02/30/90

Accounts Receivable Aging Report

Name	Current Balance	30 days	60days	90days	90+ days
Customer Number 1000 Ahmed Salem	80.00	88.00	35.00	0.00	0.00
Customer Number 1021 Tarek Fathy	0.00	98	90	130	200

شكل (٢١ - ٤)

ولانشاء هذا التقرير يتم استخدام الأمر (CREATE REPORT) كالآتي :

```
USE Customer
CREATE REPORT Aging
```

وفي هذه الحالة تظهر قوائم برنامج المساعد (Assistant) التي يتم من خلالها تحديد محتويات الأعمدة (Columns) وذلك كما هو موضح بالشكل (٢١ - ٥) .

ولعرض رقم العميل مع اسمه يستخدم الاختيار (Group) ويتم استخدام رقم العميل حقلا للتجميع بناء عليه وكتابة عنوان المجموعة (Customer Number) وبذلك يظهر هذا العنوان يليه رقم العميل مع كل عميل كما يتضح من صورة التقرير السابق ايضاحها .

	Contents	Heading	Width	Decimal	Total
1	Trim(Cust_Name)	Name	20		
2	Start_Bal	Current , Balance	9	2	N
3	Bal_30	30 days	8	2	N
4	Bal_60	60 days	8	2	N
5	Bal_90	90 days	8	2	N
6	Bal_90Plus	90 + days	8	2	N

شكل (٢١ - ٥)

ويتم طباعة التقارير المختصرة (Summary Reports) والتقارير الزمنية (Aging Reports) باستخدام البرنامج (AgeSum.prg) مع استخدام الماكرو لتحديد نوع التقرير المطلوب . ويتضح ذلك من الأوامر التالية الموجودة فى برنامج الطباعة (ARPrint.prg) .

```
CASE PChoice = 2
    RepForm = "ARSumm"
    DO AgeSumm
CASE PChoice = 3
    RepForm = "Aging"
    DO AgeSumm
```

بعد اختيار المستخدم للرقم (2) يتم تخزين كلمة (ARSumm) في متغير الذاكرة (RepForm) . وعند اختياره للرقم (3) يتم تخزين كلمة (Aging) في نفس هذا المتغير . وعند تنفيذ البرنامج (AgeSum) فى الحالتين يتم طباعة التقرير الخاص بكل حالة .

ويتم كتابة سطور هذا البرنامج كالآتى :

```
* * * * * * * * * * * * * * * * * * AgeSumm.prg
*    Print aged balances or summary reports.
*    Called from A/R Print menu.
@ 5,0 CLEAR

* - - - - -  Ask about printer
STORE "  " TO LP, Mac
@ 15,5 SAY "Send report to printer ? (Y/N)" ;
  GET LP PICT "!"
READ
* - - - - -  Set up the printer if necessary.
IF LP = "Y"
    Mac = "TO PRINT"
    WAIT "Prepare printer,then press any key to print"
ENDIF

* - - - - Use Customer database and report stored in
* - - - - RepForm
CLEAR
USE Customer INDEX CustNo
REPORT FORM & RepForm & Mac

* - - - - Pause , if necessary , then return to menu
IF LP # "Y"
    WAIT "Press any key to return to AR Print menu.
ELSE
    EJECT
INDIF
RETURN
```

وعند تنفيذ هذا البرنامج يتم التعويض بنوع التقرير المطلوب سواء كان تقريرا مختصرا (ARSumm) أو زمنيا (AgeSumm) مكان المتغير (RepForm) حسب اختيار المستخدم . كما يتم التعويض بالسلسلة الحرفية (TO PRINT) مكان المتغير (Mac) اذا اختار المستخدم الطباعة . وهذا يؤدى الى أن يصبح الأمر فى الصورة التالية :

```
REPORT FORM ARSumm TO PRINT
```

وذلك فى حالة اختيار المستخدم للاختيار رقم (2) من قائمة الطباعة . وبالتالى

يتم طباعة التقرير المطلوب .

٢١ - ٦ التقارير التاريخية (ARHist.prg)

التقارير التاريخية تسمح للمستخدم بالبحث عن أى بيانات سابقة للعميل خلال أى فترة زمنية . ويتم الدخول فى قائمة التقارير التاريخية عندما يختار المستخدم الاختيار رقم (5) من القائمة الرئيسية لبرنامج طباعة تقارير حسابات العملاء . وفى هذه الحالة تظهر القائمة الموضحة بالشكل (٢١ - ٦) .

```
History Menu                                02/20/90

        1. Search by Customer Code
        2. Search by Product Code
        3. Search by Date
        4. Return to Main Menu

  Enter Choice ( 1 - 4 )
```

شكل (٢١ - ٦)

وعندما يختار المستخدم الاختيار رقم (1) من القائمة ثم يقوم بكتابة رقم العميل المطلوب ، تظهر جميع البيانات التاريخية الخاصة بهذا العميل بالصورة الموضحة بالشكل (٢١ - ٧) . ويلاحظ من هذا الشكل أن التقرير مكون من جزأين ، جزء علوى وجزء سفلى . وهذا يتطلب انشاء صورتين للتقرير باستخدام الأمر (CREATE REPORT) . حيث يتم طباعة الجزء العلوى من التقرير باستخدام الملف (CusHist1.frm) ويتم انشاؤه عن طريق كتابة السطرين التاليين :

```
USE ChrgHist
CREATE REPORT CusHist1
```

ويلاحظ هنا استخدام الملف (ChrgHist.dbf) الذى سبق انشاؤه قبل كتابة برنامج حسابات العملاء .

وفى هذه الحالة تظهر قوائم برنامج المساعد (Assistant) التى يتم عن طريقها تحديد عنوان التقرير (Customer History) . كما يتم تحديد محتويات الأعمدة كالآتى :

(Check_No) ،(Part_No) ،(Qty) ،(Descript) ،(Unit_Price) ، (Amount) ، (Date) .

كما يتم تحديد عناوين الأعمدة كما هو واضح من الشكل (٢١ - ٧) .

Page No: 1
02/20/90
Customer History

Cust No.	Part No.	Qty	Description	Unit Price	Total	Date
1000	BBB	5	Floppy disks	1600	80	01/30/90
1000	AAA	2	Printer	900	1800	02/05/90

Cust. No.	Check No.	Description	Amount	Date
1000	1333	Payment	100.00	02/06/90
1000	1750	Payment	200.00	02/30/90

شكل (٢١ - ٧)

أما الجزء السفلى من التقرير فهو يتعلق بموقف التسديد الخاص بهذا العميل . ويتم تكوينه بواسطة الملف (CusHist2.frm) الذى يتم انشاؤه بكتابة السطرين التاليين .

```
USE PayHist
CREATE REPORT CusHist2
```

وفى هذه الحالة تظهر قوائم برنامج المساعد كما سبق الايضاح ويتم من خلالها تحديد محتوات الأعمدة كالآتى :

(Cust_No) ،(Check_No) ، (Descript) ،(Amount) ،(Date)

كما يتم تحديد عناوين هذه الأعمدة كما هو واضح من الشكل (٢١ - ٧) .

واذا اختار المستخدم الاختيار رقم (2)من قائمة التقارير التاريخية فان هذا يعنى أن التقرير المطلوب عن صنف معين . ولذلك يتم ادخال رقم هذا الصنف (Part_no) . وفى هذه الحالة يظهر التقرير الموضح بالشكل (٢١ - ٨) .

Page No. 1
02/20/90

Product Code History

Part No	Description	Qty	Unit Price	Total	Date
AAA	Printer	2	900	1800	02/05/90
AAA	Printer	3	900	2700	02/10/90
AAA	Printer	5	900	4500	02/20/90
Total		10	2700	8200	

شكل (٢١ - ٨)

ويتم انشاء هذا التقرير عن طريق كتابة السطرين التاليين :

```
USE BillHist
CREATE REPORT CodeHist
```

وعند ظهور قوائم الاختيارات الخاصة ببرنامج المساعد (Assistant) يتم كتابة عنوان التقرير (Heading) كالآتى :

(Product Code History)

كما يتم تحديد محتويات الأعمدة (Columns) كالآتى :

(Part_No)، (Descript)، (Qty)، (Unit_Price)، (Amount)، (Date)

ويفيد هذا التقرير فى متابعة موقف كل صنف ومعرفة معدل صرفه .

واذا اختار المستخدم الرقم (3) من قائمة التقارير التاريخية فان هذا يعنى أن المستخدم يريد طباعة تقرير عن موقف الموازنة فى تاريخ محدد . وفى هذه الحالة يقوم بادخال التاريخ المطلوب .

وهذا التقرير يتكون من جزأين . الجزء العلوى يتم انشاؤه بواسطة الملف (DatHist1.frm) ويتم ذلك عن طريق كتابة السطرين التاليين :

```
USE BillHist
CREATE REPORT DatHist
```

ويتم كتابة العنوان (Product Code History) للجزء العلوى ، كما يتم تحديد محتويات الأعمدة (Columns) كما يلى :

(Date)، (Part_No)، (Descript)، (Qty)، (Unit_Price)، (Amount) ، (Cust_No)

والجزء السفلى يتم انشاؤه بواسطة الملف (DatHist2.frm) . ويتم ذلك بكتابة السطرين التاليين :

```
USE PayHist
CREATE REPORT DatHist2
```

ويتم تحديد محتويات الأعمدة كالآتى :

(Date)، (Check_No) ، (Amount) ، (Descript) ، (Cust_No) .

ويجب ملاحظة أن هذه المجموعة من التقارير هي مجرد اقتراحات . ويستطيع مخطط البرامج انشاء أى تقارير أخرى حسب الحاجة . وبعد انشاء التقارير المختلفة يتم كتابة البرنامج الذى يؤدى الى طباعة كل نوع من هذه التقارير . وذلك كالآتى :

```
* * * * * * * * * * * * * * * *   ARHist.prg,
*  Search history, and current charges and payments
*  files and display summary data.
*    Called from AR Print Options menu.
HChoice = 0
DO WHILE HChoice # 4
    CLEAR
    DO Title WITH "History Menu"
    TEXT
        1. Search by customer code
        2. Search by product code
        3. Search by date
        4. Return to main menu.
    ENDTEXT
    @ 24,1 SAY "Enter choice (1 - 4) "
      GET HChoice PICT "9" RANGE 1,4
    READ
    * - - - - - - -  Set up search macro accordingly.
    @ 4,0  CLEAR
    DO CASE
       CASE HChoice = 1
           M_Cust_No = 0 ;
           @ 15,5 SAY "Enter customer number"
             GET M_Cust_No PICT "99999"
           READ
           LookAT = "Cust_No"
           LookFOR = M_Cust_No
           RepForm1 = "CusHist1"
           RepForm2 = "CusHist2"
           SET EXACT ON
       CASE HChoice = 2
```

```
        M_Code = SPACE (5)
        @ 15,5 SAY "Enter product code" GET M_Code
        READ
        LookAT = "UPPER(Part_No)"
        LookFOR = UPPER(M_Code)
        RepForm1 = "CodeHist"
        SET EXACT ON
    CASE HChoice = 3
        M_Dat = SPACE (8)
        @ 15,5 SAY "Enter Date " GET M_Date
        READ
        LookAT = "DTOC(Date)"
        LookFOR = TRIM(M_Date)
        RepFlrm1 = "DatHist1 "
        RepForm2 = "DatHist2"
        SET EXACT OFF
    CASE Hchoice = 4
        SET EXACT OFF
        RETURN
ENDCASE
* - - - - - -  Ask about printer.
@ 5,0  CLEAR
STORE "  " TO Lp, Mac
@ 15,5 SAY "Send report to printer ? (Y/N)" ;
  GET LP PICT "!"
READ
IF Lp = "Y"
    Mac = "TO PRINT
ENDIF
* - - - - - -  Search billing History file
USE BillHist
SET FILTER TO &LookAt = LookFor
COPY TO Temp
USE Temp
APPEND FROM Charges FOR & LookAT = LookFOR
SET FILTER TO &LookAT = LookFOR
REPORT FORM &RepForm1 &Mac
```

```
    * - - - if not searching for product code , Search
    * - - - Payments files
    IF HChoice # 2
        USE PayHist
        SET FILTER TO &LookAT = LookFOR
        REPORT FORM &RepForm2 &Mac PLAIN NOEJECT
    ENDIF
    * - - - IF report not going to printer, pause.
    IF Lp # "Y
        ?
        ?
        WAIT
    ENDIF
ENDDO(HChoice # 4)
```

الفصل الثانى والعشرون

التحديث الشهرى للنظام



يتم تحديث النظام مرة واحدة كل شهر عندما يختار المستخدم الرقم (6) من القائمة الرئيسية لبرنامج حسابات العملاء حيث يتم تشغيل برنامج الترحيل الشهرى للنظام (Post.prg) الذى يقوم بالآتى :

١ - نقل بيانات الموازنة خلال ثلاثين يوما ، أو ستين يوما أو ٩٠ يوما أو أكثر من ذلك في بيان الفترة السابقة على الترتيب وذلك باستخدام الأمر (REPLACE) بحيث تنتقل بيانات حقل الموازنة خلال (٣٠) يوما الى حقل الموازنة خلال (٦٠) يوما وبيان حقل الموازنة خلال (٦٠) يوما الى حقل الموازنة خلال (٩٠) يوما . وتضاف بيانات حقل الموازنة خلال (٩٠) يوما الى بيانات حقل الموازنة خلال أكثر من ٩٠ يوما .

٢ - جميع فواتير حركة الصرف التي تمت كتابتها يتم تلخيصها وادخالها في ملف قاعدة بيانات جديد يسمى (Summary.dbf) . ثم يتم تحديث الملف الرئيسي (Master.dbf) مـــن الملف (Summary.dbf) باستخدام الأمر (UPDATE FROM) .

٣- كل السجلات الخاصة بالفواتير التي تم تسديدها (Billed = .T.) تضاف الى ملف الصرف التاريخي (BillHist.dbf) وتمسح من ملف حركة الصرف (Charges.dbf) وذلك لتجهيز ملف حركة الصرف للشهر الجديد .

٤ - يتم تنفيذ نفس هذه العملية على ملف حركة التسديد (Payments.dbf) . حيث يتم نقل جميع السجلات التي تم ترحيلها (Posted = .T.) الى ملف التسديد التاريخي (PayHist.dbf) . كما يتم مسحها من ملف حركة التسديد (Payments.dbf) .

٥- بعد الانتهاء من ترحيل سجلات الفواتير التى تم تسديدها يتم تعديل محتويات حقل الموازنة الابتدائية للشهر (Starting Balance) وكذلك حقل تاريخ آخر تحديث (Last Updating) باستخدام الأمر (REPLACE) .

٦ - يقوم البرنامج أيضا باستخدام كلمة مرور (Password) لتامين عملية ترحيل البيانات حتى لاتتم بواسطة شخص غير مسئول . وقد تم استخدام كلمة (Mohamed) ككلمة مرور ولكن يمكن استخدام أى كلمة أخرى . ونظرا لأن عملية الترحيل الشهري لحركة الصرف والتسديد قد تأخذ وقتا طويلا اذا كان الملف كبيرا جدا ، لذلك يتم كتابة السطور التالية :

```
*     - - - -Show progress
@ 20,1 SAY "Aging the balances : Record" + ;
   STR(RECNO(),4) + "OF" + STR(RECCOUNT() , 4)
```

وهذه السطور عند تنفيذها فى البرنامج تؤدى الى ظهور الآتى على الشاشة :

```
Aging the balance : Record 2 of 100
```

ومع نقل كل سجل يزيد العدد التالى لكلمة (Record) بواحد حتى انتهاء عملية النقل . وهذه الرسالة تؤدى الى اطمئنان المستخدم أن عملية النقل تتم بدون مشاكل .

والبرنامج (Post.prg) يتم كتابته كالآتى :

```
* * * * * * * * * * * * * * * *Post.prg
*    Posts summarized monthly accounts to the Customer
*    file. Called from AR main menu .
SET DELETED ON
CLEAR
DO Title WITH "Monthly Posting"
* - - - - -  Display and get password
TEXT

   This is the program to post Payments and charges.
   Be sure you have printed all the monthly invoices
   before proceeding with this program.

ENDTEXT
Password = SPACE(7)
@ 15,12 SAY "Enter password to proceed" GET Password ;
        PICT "!!!!!!!"
READ

* - - - - If proper password not entered return to the
* - - - - - - -  menu.
IF Password # "MOHAMED"
    ? "Illegal password" , CHR(7)
    SET DELETED OFF
    RETURN
ENDIF
```

```
* - - Do the posting,first,shift all current 30 , 60
* - - and 90 day billings "back" one field in the
* - - customer file.
@ 20,1 SAY "Working ...."
CLOSE DATABASES
USE Customer;
REPLACE ALL;
    Bal_90Plus WITH Bal_90Plus + Bal_90, ;
    Bal_90 WITH BAL_60, ;
    BAL_60 WITH Bal_30, ;
    Bal_30 WIHT Chg_Curr - Pay_Curr
REPLACE ALL; ,
    Chg_Curr WITH 0 , ;
    Pay_Curr WITH 0

* - - - Now,create summary of the charges database by
* - - - customer number.
SET SAFETY OFF
USE Charges INDEX ChrgNo
COPY STRUCTURE TO Summary
TOTAL ON Cust_No TO Summary FIELDS Qty , ;
  Unit_Price ,Amount FOR Billed
* - -Now update the customer database current balances
* - -with data from the charges summary file.
SELECT 1
USE Customer INDEX CustNo
SELECT 2
USE Summary
SELECT 1
UPDATE ON Cust_No FROM Summary REPLACE Chg_Curr WITH ;
   B -> Amount

* - - - Move all posted transactions to the billing
* - - history file
SELECT 2
USE BillHist
APPEND FROM Charges FOR Billed
```

```
* - - - - - -  Then empty the current charges file.
CLOSE DATABASE
USE Charges INDEX ChrgNo
DELETE ALL FOR Billed
PACK

* - - Now , summarize payment totals for each Customer
USE Payments INDEX PayNo
COPY STRUCTURE TO Summary
TOTAL ON Cust_No TO Summary Fields Amount FOR Posted

* - -Now update the customer database current balances
* - -with data from the payments summary file.
SELECT 1
USE Customer INDEX CustNo
SELECT 2
USE Summary
SELECT 1
UPDATE ON Cust_No FROM Summary REPLACE Pay_Curr ;
   WITH B -> Amount

* - - Append all posted transactions to the payments
* - - history file.
SELECT 2
USE PayHist
APPEND FROM Payments FOR Posted

* - - - - - -  Then empty the current payments file.
CLOSE DATABASES
USE Payments INDEX PayNo
DELETE ALL FOR Posted
PACK

* - - - Then update the 'last billed' and 'starting
* - - balance' Fields in the customer database.
USE Customer
REPLACE ALL Start_Bal WITH Start_Bal + Chg_Curr ;
  - Pay_Curr
```

```
REPLACE ALL Last_Updat WITH DATE()

* - - - - -  Adjust aged balances.
GO TOP
DO WHILE .NOT. EOF()
    * - - - - -  Show progress.
    @ 20,1 SAY "Aging the balances : Record" + ;
       STR(RECNO(), 4) + "OF" + STR(RECCOUNT(), 4)
    * - - - - - -  IF no payment, skip calculations.
    IF Pay_Curr < = 0
        SKIP
        LOOP
    ENDIF (Pay_Curr < = 0)
    * - - - - - -  Otherwise , subtract the payment
    More = .F.
    NextBal = .T.
    IF Bal_90Plus > 0
        Remain = Pay_Curr - Bal_90Plus
        IF Remain > = 0
            REPLACE Bal_90Plus WITH 0
            More = .T.
        ELSE
            REPLACE Bal_90Plus WITH ABS(Remain)
            NextBal = .F.
        ENDIF(Remain > = 0)
    ENDIF(90+ > 0)
    * - - - - - -  90 days.
    IF NextBal .AND. Bal_90 > 0
        IF More
            Remain = Remain - Bal_90
        ELSE
            Remain = Pay_Curr - Bal_90
        ENDIF(More)
        IF Remain > = 0
            REPLACE Bal_90 WITH 0
            More = .T.
        ELSE
            REPLACE Bal_90 WITH ABS(Remain)
```

```
            NextBal = .F.
        ENDIF(Remain >= 0)
    ENDIF (NextBal & Bal90 > 0)
    * - - - - -  60 days.
    IF NextBal .AND. Bal_60 > 0
        IF More
            Remain = Remain - Bal_60
        ELSE
            Remain = Pay_Curr - Bal_60
        ENDIF(More)
        IF Remain > = 0
            REPLACE Bal_60 WITH 0
            More = .T.
        ELSE
            REPLACE Bal_60 WITH ABS(Remain)
            NextBal = .F.
        ENDIF(Remain >= 0)
    ENDIF (NextBal & Bal60 > 0)
    * - - - - - -  30 days.
    IF NextBal .AND. Bal_30 > 0
        IF More
            Remain = Remain - Bal_30
        ELSE
            Remain = Pay_Curr - Bal_30
        ENDIF(More)
        IF Remain > = 0
            REPLACE Bal_30 WITH 0
            More = .T.
        ELSE
            REPLACE Bal_30 WITH ABS(Remain)
            NextBal = .F.
        ENDIF(Remain >= 0)
    ENDIF (NextBal & Bal30 > 0)
SKIP
ENDDO (end of file)
```

```
* - - - - - -  Display closing messages
CLEAR
? CHR(7)
TEXT

    The posting procedure is complete. Use option 4,
    from the main menu to print current monthly
    summary and aging reports.

ENDTEXT

* - - - - - -  Get rid of any old keypresses.
CLEAR TYPEAHEAD
WAIT "Press any key to return to main menu ...."
* - - - - - -  Return to the main menu.
SET DELETED OFF
CLOSE DATABASES
RETURN
```

ويلاحظ في البرنامج استخدام عدة أوامر (IF) فى تنفيذ الحسابات الخاصة بكل حالة . فمثلا اذا لم يتم تسديد أى قيمة جديدة أى (Pay_Curr <= 0) ، فان هذا يعني أنه ليست هناك حسابات مطلوب اجراؤها . ولذلك يتم الانتقال الى العميل التالى . وخلاف ذلك يتم ضبط حسابات (٣٠) يوما ، (٦٠) يوما، (٩٠) يوما ، وأكثر من ٩٠ يوما لكل عميل .

ويتم استخدام المتغير (Remain) فى عملية الضبط . فاذا كان المبلغ الذى تم تسديده يغطى الحسابات القديمة أكثر من ٩٠ يوما، فى هذه الحالة يتم تصفير هذه الحسابات (جعلها تساوى صفرا) . وهكذا يتم المرور على باقى الحسابات (٩٠ يوما ، (٦٠) يوما ، (٣٠) يوما على الترتيب .

وعند الانتهاء من حساب جميع العملاء يتم عرض رسالة للمستخدم عن التقارير المطلوبة . ثم يتم اغلاق جميع الملفات والعودة الى القائمة الرئيسية .



الباب الثالث والعشرون

برنامج التكامل بين حسابات العملاء والمخازن



يستخدم هذا البرنامج في ربط برنامج حسابات العملاء ببرنامج المخازن. ولتنفيذ ذلك يجب اجراء التعديلات التالية :

١ - يتم طرح كميات الأصناف التي تم ادخالها في حركة الصرف من كميات الأصناف الموجودة في ملف المخازن الرئيسي (Master.dbf).
٢ - يتم تمييز السجلات التي يتم ترحيلها الى ملف المخازن الرئيسي (Master.dbf) بعلامة معينة حتى لايتم ترحيلها مرة أخرى.
٣ - يتم اختبار حركة الصرف (Charges) في برنامج حسابات العملاء أو حركة السداد (Payemnts) قبل ترحيلها الى الملف الرئيسي للمخازن .

وفي البداية يجب اضافة حقل جديد الى ملف حركة الصرف (Charges.dbf) يوضح للبرنامج اذا كان السجل قد تم ترحيله الى ملف المخازن أم لا . وهذا الحقل يكون حقلا منطقيا ونسميه (InPost) حيث يمثل الحرفان (In) أول حرفين في كلمة (Inventory) ويتم ذلك باستخدام الأمر (MODIFY STRUCTURE). كمايتم اضافة نفس الحقل الى ملف الصرف التاريخي (BillHist.dbf) حتى يصبح بنفس التركيب . حيث أن هذا الملف يستقبل السجلات التي يتم ترحيلها الى ملف حسابات العملاء الرئيسي (Customer.dbf).

ولتامين عملية الترحيل يجب التأكد أن كل سجل جديد يتم اضافته الى ملف حركة الصرف (Charges.dbf) يتم تمييزه حتى يعلم البرنامج أنه لم يتم ترحيله الى ملف المخازن الرئيسي (Master.dbf). ويتم ذلك عن طريق اضافة سطر معين الى برنامج اضافة العملاء (NewChrg.prg) في الجزء الخاص باضافة السجل الجديد. حيث يتم اضافة السطر التالي :

```
REPLACE InPost WITH .F.
```

ويصبح هذا الجزء من البرنامج كالآتي :

```
IF .NOT. Exiting
    SELECT 2
    APPEND BLANK
    REPLACE Cust_No WITH M_Cust_No
    REPLACE Date WITH DATE()
    REPLACE Billed WITH .F.
    REPLACE InPost WITH .F.
    SET FORMAT TO FNewChrg
    READ
    CLOSE FORMAT
    REPLACE Amount WITH Qty * Unit_Price
ENDIF
```

ثم يتم كتابة البرنامج الذى يقوم بتحديث ملف المخازن الرئيسي من كل السجلات التى لم يتم ترحيلها (InPost=.F.) من ملف حركة الصرف (Charges.dbf) وملف الصرف التاريخي(BilHist.dbf) كما يقوم بتغيير حقل التحديث (Inpost) الى (True) حتى لايتم ترحيله مرة ثانية .

وحيث أن البرنامج (Updater.prg) هو البرنامج الذى يقوم بتحديث ملف المخازن الرئيسي . لذلك يتم اضافة سطر الى هذا البرنامج يؤدى الى تشغيل برنامج آخر اسمه (ARUpdate.prg) يختص بتحديث الملف الرئيسي من ملف حركة الصرف . ويتم اضافة هذا السطر قبل نهاية ملف التحديث (Updater) كالآتى مثلا :

```
DO ARUpdate
```

والبرنامج (ARUpdate.prg) يتكون من السطور التالية :

```
* * * * * * * * * * * * * * * *ARUpdate.prg
*   Update the master file from charges and BilHist.
CLEAR
? "Updating from the A/R System .."
* - - - - -  Use the charges databases.
USE Charges
* - - - - -  Copy nonupdated records to Temp File.
COPY STRUCTURE TO Temp
COPY TO Temp FOR .NOT. InPost
* - - - - - -  Now get the recrods from BillHist.
USE Temp
APPEND FROM BillHist FOR .NOT. InPost
* - - - - - -  Get the Temp file sorted by part number.
INDEX ON Part_No TO ARIndex

* - - - - - -  Use the Master file for updating .
SELECT 1
USE Master INDEX Master
SELECT 2
USE Temp INDEX ARIndex
```

```
* - - - Update Master from the temporary A/R file.
SELECT 1
UPDATE ON Part_No FROM Temp REPLACE Qty WITH ;
    Qty - Temp -> Qty

* - - - Use the origingal charges database file
* - - - change all posted fields to true
CLOSE DATABASES
USE Charges
REPLACE ALL InPost WITH .T.

* - - - - - - -  Do the same thing with BillHist file.
USE BillHist
REPLACE ALL InPost WITH .T.
* - - - - - -  Return to the updater command file.
RETURN
```



5

الجزء الخامس

بعض الادوات المتقدمة

* برنامج كتابة الشيكات
* برنامج اختيار الألوان
* برنامج تحريك العمود الضوئى



مقدمة

هذا الجزء يقدم مجموعة من الأدوات المتقدمة (Advanced Tools) التي يمكن لمخطط البرامج استخدامها في كثير من التطبيقات . وهذه الأدوات بالاضافة الى ماتوفره للمستخدم من جهد ووقت ، فانها أيضا تزيد من كفاءة البرنامج الذى يجرى اعداده .

و يتكون هذا الجزء من ثلاثة فصول . الفصل الأول يشرح برنامج يمكن المستخدم من طباعة الشيكات مع القدرة على كتابة مبلغ الشيك بالأرقام و الحروف . و الفصل الثاني يتيح لمخطط البرامج استخدام الألوان في الشاشات و القوائم التي يتم عرضها على المستخدم . كما يتيح للمستخدم اختيار الألوان المناسبة له من خلال قائمة اختيارت خاصة . و الفصل الثالث يتيح لمخطط البرامج تصميم قوائم اختيارات تسمح للمستخدم بتحريك مؤشر على الشاشة الى الاختيار المطلوب ، و ذلك علاوة على الطريقة التقليدية في الاختيار عن طريق الأرقام أو الحروف .


Converted by Tiff Combine - (no stamps are applied by registered version)

الفصل الرابع و العشرون
برنامج كتابة الشيكات



في معظم الأحيان يحتاج مخطط البرامج الى طباعة الشيكات من خلال البرنامج . وعادة ما يحتوى الشيك على الرقم الممثل للمبلغ المطلوب صرفه ، كما يحتوى أيضا على ترجمة انجليزية لهـــذا الرقم . فمثلا الرقم (950) يتم كتابته كالآتى :

(Nine Hundred and Fifty)

ولكن الأعداد التى يتم الحصول عليها من برنامج قاعدة البيانات تكون دائما على الصورة العددية .

وحيث أن البرنامج لايعرف مقدما القيمة العددية التى يتم كتابتها فى الشيك . لذلك يصبح فى منتهى الصعوبة على مخطط البرامج تحويل كل عدد الى الصورة الحرفية عند كتابته فى الشيك . ولذلك فان هذا البرنامج يتيح لمخطط البرامج ترجمة كل عدد من (1) حتى (999,999.99) الى الأعداد الحرفية المقابلة .

ولتوضيح ذلك يتم أولا انشاء ملف قاعدة بيانات لكتابة الشيك حتى يتم عن طريقه اختبار البرنامج بعد ذلك . ويتكون هذا الملف من الحقول التالية :

Field	Field Name	Type	Width	Dec
1	CHECK_NO	Numeric	5	0
2	TO_WHOM	Character	25	
3	AMOUNT	Numeric	9	2
4	DATE	Date	8	
** TOTAL **			48	

شكل (٢٤ - ١)

وقبل كتابة البرنامج يجب اولا تخزين المقابل الحرفى لكل عدد من الأعداد من واحد الى عشرين فى متغيرات ذاكرة وكذلك أرقام العشرات مثل (Thirty) ، (Fourty) ، .... الخ . كما يتم تخزين هذه المتغيرات فى ملف ذاكرة (Memory File) وهذا الملف نسميه (English.mem) ويتم انشاؤه من خلال البرنامج التالى :

```
* * * * * * * * * * * * * * * * * *English.prg
* sets up memory file for storing English equivalents.
CLEAR
? "Creating English.mem file with English for numbers"
?
SET DEFAULT TO C
SET TALK ON
CLEAR MEMORY

U   = " "
U1  = "ONE"
U2  = "TWO"
U3  = "TREE "
U4  = "FOUR "
U5  = "FIVE "
U6  = "SIX "
U7  = "SEVEN "
U8  = "EIGHT "
U9  = "NINE "
U10 = "TEN "
U11 = "ELEVEN "
U12 = "TWELVE "
U13 = "THIRTEEN "
U14 = "FOURTEEN "
U15 = "FIFTEEN "
U16 = "SIXTEEN "
U17 = "SEVENTEEN"
U18 = "EIGHTEEN"
U19 = "NINETEEN"
U20 = "TWENTY"
```

```
U30 = "THIRTY"
U40 = "FOURTY"
U50 = "FIFTY"
U60 = "SIXTY"
U70 = "SEVENTY"
U80 = "EIGHTY"
U90 = "NINETY"

* - - - - -  Save all variables to English.mem file .
SAVE TO English
CLEAR
?
?
?
SET TALK OFF
RETURN
```

وهذا البرنامج يؤدى الى انشاء ملف الذاكرة (English.mem) الذى يجب تحميله فى الذاكرة عند تشغيل برنامج انشاء الشيكات . ويلاحظ أن كل متغير يبدأ اسمه بالحرف (U) يليه رقم يمثل الرقم المطلوب تحويله فمثلا (U5) يمثل (Five) ، (U30) يمثل (Thirty) . وهكذا .

ولتحميل هذا الملف فى الذاكرة يتم كتابة السطر التالى :

```
RESTORE FROM English
```

والخطوة التالية بعد ذلك هى انشاء ملف الخطوات (Procedure File) أو البرنامج الذى يقوم بترجمة أى عدد الى المقابل الحرفى له . وهذا البرنامج نسميه (Translat.prg) ويتم كتابته كالآتى :

```
* * * * * * * * * * * * * * * * *Translat.prg
* - - - - -  Procedure to convert a number to English
* - - - - -  equivalent.
PROCEDURE Translat
PARAMETERS Number, English
```

```
*    Set up memory variables
Counter = 1
Start = 1
String = STR(Amount, 9, 2)
* - - - - - -  Loop through thousands and hundreds
DO WHILE Counter < 3
    * - - - Split out hundreds , tens and ones.
    Chunk = SUBSTR(String, Start , 3)
    Hun = SUBSTR(Chunk , 1, 1)
    Ten = SUBSTR (Chunk, 2, 2)
    One = SUBSTR(Chunk , 3, 1)

    * - - - - - -  Handle hundreds portion.
    IF VAL(Chunk) > 99
        English = English + U & Hun + "Hundreds"
    ENDIF
    * - - - - - -  Handle second 2 digits.
    T = VAL(Ten)
    IF T > 0
        DO CASE
          * - - - Case 1 : handle even tens and teens;
          CASE (INT(T/10.0) = T/10.0) .OR. (T > 9 ;
            AND. T < 20)
            English = English + U & Ten
            * - - Case 2 : Handle greater than 10 but
            * - - - not evenly divisable.
          CASE T > 9 .AND. (INT(T/10.0) # T/10.0)
            Ten = SUBSTR(Ten, 1,1) + '0'
            English = English + U & Ten + " " + U&one
            * - - - Case 3 : Handle less than 10
          CASE T < 10
            English = English + U & One
       ENDCASE
    ENDIF (T > 0)
    * - - - - - -  Add "Thousand" if necessary
```

```
    IF Amount > 999.99 .AND. Counter = 1
        English = English + "THOUSAND"
    ENDIF (need to add "Thousand")
    * - - - - -  Prepare for pass through hundreds.
    Start = 1
    Counter = Counter + 1
ENDDO (while counter < 3)
* - - - - -  Take out ratios
IF INT(Amount) > 0
    English = English + "And"
ENDIF
English = English + SUBSTR(String, 8 , 2) + "/100"
```

ويمكن استخدام هذا البرنامج داخل أى برنامج آخر عن طريق كتابة السطرالتالى :

```
SET PROCEDURE TO Translate
```

ويؤدى هذا الى فتح ملف الخطوات السابق . ويجب قبل تشغيل البرنامج تعريف المعاملات التى سوف تستخدم معه . وذلك كالآتى :

```
M_Amount = 9845
English = "  "
DO Translat WITH M_Amount , English
```

ويجب قبل ذلك فتح ملف الذاكرة (English) وذلك بكتابة السطر التالى :

```
RESTORE FROM English
```

وبذلك يمكن اختبار البرنامج . ولكتابة البرنامج الذى يؤدى الى طباعة الشيك بالكامل يتم كتابة السطور التالية :

```
* * * * * * * * * * * * * *Checks.prg
* - - - - -  Sample program to write checks
```

```
CLEAR
SET TALK OFF
SET SAFETY OFF
SET PROCEDURE TO Translat
* - - - - - -  Bring English equivalent variables.
RESTORE FROM English
* - - - - - -  Use the checks database
USE Checks
INDEX ON Check_No TO CheckNo
?
? "First Check to be printed will be", Check_No
?
?
WAIT "Press any key to begin writing checks"
CLEAR
SET PRINT ON
GO TOP
DO WHILE .NOT. EOF()
    * - - - - - -  Translate Amount to English
    M_Amount = Amount
    English = "  "

    DO Translat WITH M_Amount , English
    * - - - Print the check with required modification
    * - - - - - -  for the check
    ? "                    " Date
    ?
    ? To_Whom, "    " , Amount
    ?
    ? English
    ?
    ?
    ?
    SKIP
ENDDO (not eof)
```

```
SET PRINT OFF
CLOSE PROCEDURE
CLEAR MEMORY
CLOSE DATABASES
RETURN
```

هذا البرنامج يؤدى الى طباعة شيكات تحتوى على المبلغ (Amount) مكتوبا بالأرقام وكذلك بالحروف .

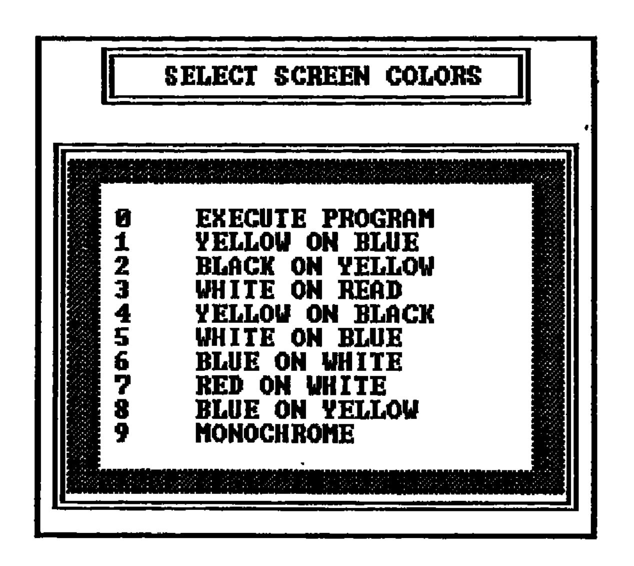


الفصل الخامس والعشرون

برنامج اختيار الالوان



هذا البرنامج يسمح للمستخدم باختيار الوان الشاشة وكذلك ألوان الأعمدة الضوئية (Highlights) . وذلك عن طريق عرض قائمة بالألوان المختلفة والسماح للمستخدم باختيار الألوان المطلوبة أنظر الشكل (٢٥ - ١).



شكل (٢٥ - ١)

ويمكن تسمية هذا البرنامج (Colors.prg) ويتم كتابته كالآتى :

```
* * * * * * * * * * * * * * *Colors.prg
* - - - - -  to give the user the colors he needs.
CLEAR
SET TALK OFF
SET STATUS OFF
* - - - - -  Set up loop for displaying the menu
DO WHILE .T.
    * - - - - -  Drow a box
    @ 1,24 TO 3,50 DOUBLE
    * - Set up loop for drawing a frame inside the box
    R = 6
```

```
    DO WHILE R > 5 .AND. R < 20
        @ R,22 SAY REPLICATE(CHR(178),31)
        R = R + 1
    ENDDO
    * - - - - -  Erase an area from the box
    @ 7, 24 SAY CLEAR TO 18,50
    @ 5, 21 TO 20,53 DOUBLE
    * - - - - -  Display the menu."
    @ 2, 28 SAY "Select Screen Colors"
    @ 8, 25 SAY " 0 - Execute Program"
    @ 9, 25 SAY " 1 - Yellow On Blue"
    @ 10,25 SAY " 2 - Black On Yellow"
    @ 11,25 SAY " 3 - White On Red"
    @ 12,25 SAY " 4 - Yellow On Black"
    @ 13,25 SAY " 5 - White On Blue"
    @ 14,25 SAY " 6 - Blue On White"
    @ 15,25 SAY " 7 - Red On White"
    @ 16,25 SAY " 8 - Blue On Yellow"
    @ 17,25 SAY " 9 - Monochrom
WAIT "   " TO P

DO CASE
    CASE P = 0
        RETURN
    CASE P = 1
        COLSTR = "GR + /B, W/R, GR"
    CASE P = 2
        COLSTR = "N/GR, W/R, GR"
    CASE P = 3
        COLSTR = "W/R , W/N"
    CASE P = 4
        COLSTR = " GR + /N , W/R , GR"
    CASE P = 5
        COLSTR = " W/B , W/R , B"
    CASE P = 6
        COLSTR = " B/W , W/R , R"
    CASE P = 7
        COLSTR = "R/W , W/N , B"
```

```
    CASE P = 8
        COLSTR = "B/GR , W/R , B"
    CASE P = 9
        COLSTR = "7/0 , 0/7 , 0"
ENDCASE

SET COLOR TO & COLSTR
CLEAR
ENDDO
```

ويلاحظ في بداية البرنامج استخدام حلقة تكرارية لرسم مستطيل داخلى وملؤه بالتظليل باستخدام الحرف CHR(178) ، ثم استخدام الأمر (CLEAR) بعد ذلك لمسح الجزء الداخلى من هذا المستطيل حتى يظهر كاطار حول قائمة الاختيارات .

كما يلاحظ أن الألوان التى تظهر على الشاشة (COLSTR) تعتمد على الرقم المخزن فى متغير الذاكرة (P) والذى تم اختياره من قائمة الألوان . وهذا البرنامج يمكن استخدامه مع أى برنامج آخر بكتابة السطر التالى :

```
DO Colors
```



الفصل السادس والعشرون

برنامج تحريك العمود الضوئى



هذا البرنامج يستخدم عندما يريد مخطط البرامج اضافة مزيد من الاثارة والتشويق الى قوائم الاختيارات . حيث يمكن عند تنفيذه عرض عمود ضوئي (Highlight) يمكن تحريكه بواسطة مفاتيح الاتجاهات لأعلى ولأسفل ( ) للوصول الى الاختيار المطلوب ثم الضغط على مفتاح الادخال لتنفيذ هذا الاختيار . وفي نفس الوقت يتيح للمستخدم الاختيار عن طريق كتابة الرقم الموجود عند أى اختيار .

وعند تنفيذ هذا البرنامج تظهر الشاشة التالية :

Sample Menu

1. Add new records
2. Print reports
3. Edit data
4. Exit

Highlight option with ↓ or ↑ and press ↵
or press appropriate menu number

شكل (٢٦ - ١)

والسطور التالية توضح نموذجا لبرنامج (Sample.prg) يستخدم هذه الطريقة فى عرض القائمة الرئيسية :

```
* * * * * * * * * * * * * * * * *Sample.prg
*    Create a sample menu with moving "light bar".
CLEAR
SET TALK OFF
```

```
* - - - - - -  create menu options (opt1-opt4)
opt1 = "1. Add new records"
Opt2 = "2. Print reports"
Opt3 = "3. Edit data"
Opt4 = "4. Exit"

* - - - - - -  Display the menu
@ 1,1  TO 3,79 DOUBLE
@ 2,32 SAY "Sample menu"
@ 5,30 SAY Opt1
@ 6,30 SAY Opt2
@ 7,30 SAY Opt3
@ 8,30 SAY Opt4

* - - Display instructions with graphics characters.
@ 14,1 TO 18,78 DOUBLE
@ 15,18 SAY "Highlight option by using"
@ 15,40 SAY CHR(24) + "or" + CHR(25) + "and press" ;
    + CHR(17) + CHR(217)
@ 17,22 SAY "or press appropriate menu number"

* - - - - - -  Initialize memory variables.
Opt = 1
Sub = STR(Opt,1)
KeyPress = 0
Choice = 0
* - - - - - -  Reverse video on option 1
@ 5,30 GET Opt1
CLEAR GETS

* - - - - - -  Loop for choosing menu options.
DO WHILE Choice # 4
    * - - - - - -  Wait for a keyPress.
    KeyPress = 0
    DO WHILE KeyPress = 0
        KeyPress = INKEY()
    ENDDO (KeyPress)
```

```
    * - - - - - -  Arrow key pressed.
    IF KeyPress = 24 .OR. KeyPress = 5
        @ Opt+4,30 SAY Opt&Sub
        Opt = IIF(KeyPress = 24, Opt+1, Opt - 1)
        Opt = IIF(Opt > 4 , 1, Opt)
        Opt = IIF(Opt < 1 , 4, Opt)
        Sub = STR(Opt,1)
        @ Opt+4,30 GET Opt &Sub
        CLEAR GETS
        LOOP
    ENDIF

    * - - - - - -  Option numbers entered.
    IF KeyPress >= 49 .AND. KeyPress <= 52
        Choice = KeyPress - 48
    ENDIF

    * - - - - - -  Return pressed
    IF KeyPress = 13
        Choice = opt
    ENDIF
    * - - - - - -  An option was selected.
    IF Choice > 0
        CLEAR
        ? 'Choice' , Choice
        Choice = 5
    ENDIF
ENDDO(Choice)
```

والبرنامج يبدأ بتخزين كل سطر من سطور القائمة فى متغير ذاكرة وذلك يتضح من السطور التالية :

```
* - - - - - create menu options (opt1-opt4)
opt1 = "1. Add new records"
Opt2 = "2. Print reports"
Opt3 = "3. Edit data"
Opt4 = "4. Exit"
```

والجزء التالى من البرنامج يؤدى الى ظهور قائمة الاختيارات على الشاشة ويتكون من السطور التالية :

```
* - - - - - Display the menu
@ 1,1  TO 3,79 DOUBLE
@ 2,32 SAY "Sample menu"
@ 5,30 SAY Opt1
@ 6,30 SAY Opt2
@ 7,30 SAY Opt3
@ 8,30 SAY Opt4
```

والجزء التالى من البرنامج يتم عن طريقه عرض لوحة المساعدة (Help) التى تظهر على الشاشة لتوضح للمستخدم كيفية الاختيار من القائمة . ويلاحظ استخدام حروف الرسم (Graphic Characters) التى يتم الحصول عليها باستخدام الدالة (CHR) . حيث يتم كتابة السهم العلوى (   ) باستخدام الداله CHR(24) ويتم كتابة السهم لأسفل باستخدام الدالة CHR(25) .

ويتم استخدام الداله CHR(17) فى كتابة حروف رسم معينة تمثل شكل مفتاح الادخال (   ) ، وهكذا . ويتضح ذلك من السطور التالية :

```
@ 14,1 TO 18,78 DOUBLE
@ 15,18 SAY "Highlight option by using"
@ 15,40 SAY CHR(24) + "or" + CHR(25) + "and press" ;
   + CHR(17) + CHR(217)
@ 17,22 SAY "or press appropriate menu number"
```

وفى الجزء التالى يتم انشاء مجموعة من متغيرات الذاكرة مثل المتغير (Opt) الذى يتم فيه تخزين الرقم الذى يتم اختياره ، والمتغير (Sub) الذى يتم فيه تخزين القيمة الحرفية لهذا الرقم ، والمتغير (KeyPress) الذى يتم فيه تخزين كود الآسكى الخاص بالمفتاح الذى يضغط عليه المستخدم ، والمتغير (Choice) الذى يتم فيه تخزين الاختيار المطلوب لاستخدامه بعد ذلك فى التفرع الى البرنامج

الخاص به . ويتم ذلك من خلال السطور التالية :

```
Opt = 1
Sub = STR(Opt,1)
KeyPress = 0
Choice = 0
```

والجزء التالى من البرنامج يؤدى الى ظهور عمود ضوئى فى المكان الممثل للاختيار رقم (1) فى القائمة وذلك باستخدام الأمر (GET) . كما يتم مسح المتغير (GET) الذى يتكون نتيجة لذلك مع بقاء العمود الضوئى على هذا الاختيار . ويتم ذلك من خلال السطرين التاليين :

```
@ 5,30 GET Opt1
CLEAR GETS
```

والجزء التالى يتم من خلاله تكوين حلقة تكرارية لعرض القائمة دائما على الشاشة حتى يختار المستخدم الخروج . وذلك من خلال السطر التالى :

```
DO WHILE Choice # 4
```

كما تستخدم حلقة تكرارية اخرى لانتظار ضغط المستخدم على اى مفتاح . وفى هذه الحالة يتم تخزين كود الآسكى (ASCII Code) الخاص بهذا المفتاح فى متغير الذاكرة (KeyPress) . وتستخدم الداله INKEY() فى الحصول على كود الآسكى الخاص بآخر مفتاح قام المستخدم بالضغط عليه . ويتضح ذلك من السطور التالية :

```
KeyPress = 0
DO WHILE KeyPress = 0
    KeyPress = INKEY()
ENDDO (KeyPress)
```

وعند ضغط المستخدم على مفتاح السهم لأعلي (CHR(24)) أو مفتاح السهم لأسفل (CHR(25)) فان ذلك يؤدى الى اختفاء العمود الضوئي من الاختيار رقم (١) . وذلك لأن الأمر (SAY) يؤدى الى اعادة كتابة السطر فوق العمود الضوئي . كما يؤدى الى اضافة واحد أو طرح واحد من الرقم المخزن فى المتغير (Opt) . ويلاحظ هنا استخدام الدالة (IIF) فى زيادة قيمة المتغير (Opt) أو انقاصه حسب القيمة المخزنة فى المتغير (KeyPress) نتيجة ضغط المستخدم على مفتاح معين . ويتضح ذلك من السطور التالية :

```
IF KeyPress = 24 .OR. KeyPress = 25
    @ Opt+4,30 SAY Opt&Sub
    Opt = IIF(KeyPress = 24, Opt+1, Opt - 1)
    Opt = IIF(Opt > 4 , 1, Opt)
    Opt = IIF(Opt < 1 , 4, Opt)
    Sub = STR(Opt,1)
    @ Opt+4, 30 GET Opt&Sub
    CLEAR GETS
    LOOP
ENDIF
```

وعندما يريد المستخدم الاختيار بالرقم وليس عن طريق تحريك العمود الضوئى فانه يكتب رقما بين (1) ، (4) . وحيث أن كود الآسكى الخاص بالأرقام يبدأ من (48) لذلك فان قيمة العدد الذى يختاره المستخدم يمكن حسابها بطرح العدد (48) من العدد الممثل لكود الآسكى (ASSII) الخاص بهذا العدد . ويتم ذلك من خلال السطور التالية :

```
* - - - - - Option numbers entered.
IF KeyPress > = 49 .AND. KeyPress < = 52
    Choice = KeyPress - 48
ENDIF
```

وعندما يضغط المستخدم على مفتاح الادخال (CHR(13)) يتم تخزين العدد الموجود فى المتغير (Opt) فى المتغير (Choice) . ويتم ذلك من خلال السطور التالية :

```
IF KeyPress = 13
    Choice = opt
ENDIF
```

وسواء كتب المستخدم رقم الاختيار المطلوب أو استخدم مفاتيح الاتجاهات فى تحريك العمود الضوئى ثم ضغط على مفتاح الادخال فان المتغير (Choice) يتم فيه تخزين رقم معين يمثل هذا الاختيار .

وفى برنامج القائمة العادية يتم استخدام الأمر (DO CASE) فى التفرع الى برنامج

معين بناء على الرقم الموجود فى المتغير (Choice) . أما فى هذا البرنامج فقد تم الاكتفاء بعرض الرقم الموجود فى المتغير (Choice) حتى يتم اختبار البرنامج والتأكد من تحقيقه للمطلوب .



هذا الكتاب

تطبيقات نظم إدارة قواعد البيانات